PLATO terminal resident specification

12/20/83

PLATO terminal resident specification

1. Introduction (11/30/83)

Notice: this is a preliminary draft and is not yet complete.

This document describes the ASCII PLATO host-terminal communications protocol.

It outlines the general principles of writing a PLATO resident (a program which communicates with the host system) and how hardware of differing capabilities might be programmed for use with PLATO.

The interface between the resident and the Micro PLATO system (which runs in the machine with the resident and makes calls upon it) is also described. If Micro PLATO is to be used with the resident, care must be taken in writing the low-level routines so that both the host communications code and the Micro PLATO interface code can easily call them.

2. Applicable documents (11/21/83)

EIA RS-232C	Interface between data terminal epuipment and data communication equipment employing serial binary data interchange.
CDC-STD 1.80.000	Programming Standards
CDC-STD 1.01.103	Software development documentation
CDC-STD 1.01.105	External reference specification
CDC-SPEC 16042866	ASCII IST-II Equipment Spec.
CDC-SPEC 16042871	ASCII IST-II ROM Resident ERS
CDC-SPEC 16042872	ASCII IST-III PLATO ERS
CDC-SPEC 16042873	ASCII IST-III Graphics ERS
CDC-SPEC 16042874	ASCII IST-III Timeshare ERS
CDC 62940006	CDC 40003-22/23 IST-III Op. Guide
CDC 62940007	CDC 40003-22/23 IST-III Hardware Maintenance Manual
CDC-SPEC 16042886	Viking X Equipment Specification
CDC-SPEC 16042887	Viking X Graphics Option
CDC-SPEC 16042888	Viking X ROM Resident ERS
CDC-SPEC 16042889	Viking X Dual Asynchronous Serial Interface
CDC-SPEC 16042890	Viking X Parallel Interface

3. Terminal-host communications (03/30/83)

3.1. General overview (10/18/83)

3.1.1. The PLATO terminal model (11/01/83)

The ASCII PLATO communications protocol assumes a certain terminal model. A particular terminal may not have all the capabilities of this model, and the resident must compensate in software for the differences between the actual hardware and the PLATO terminal model. The following sections describe features of the model and how they might be dealt with in a machine that lacks those features.

A summary of the characteristics of terminals which have been adapted for use with PLATO is included in Appendix E.

3.1.1.1. Screen resolution (11/18/83)

The PLATO terminal model has a screen resolution of 512 by 512 pixels with a square (1:1) aspect ratio. All screen coordinates are based on this.

The screen has 32 lines of 64 characters each.

The coordinate (0,0) is assumed to be at the lower lefthand corner of the screen. The coordinate (511,511) is at the upper righthand corner.

A terminal which has only a 256 vertical by 512 horizontal screen with a 2:1 aspect ratio would scale all vertical coordinates by dividing all Y coordinates by two. A terminal with 384 vertical by 512 horizontal and a 3:4 aspect ratio would multiply Y coordinates by 3/4. If a terminal has a resolution of some odd size, say, 720h by 400v and a 3:4 aspect ratio, scaling by 400/512 would not be appropriate. It would be best to scale by 3/4 and only use the center 512h by 384v area in the center of the screen.

Distortion may be unavoidable on some terminals in order to obtain the entire 512 pixel horizontal resolution. This is the recommended horizontal resolution. With it, 64 characters of eight pixels width can fit across the screen. If this is not possible due to the limitations of the target hardware, a method of scrolling across the screen to view a portion of the entire display might be considered.

3.1.1.2. Character sets (11/30/83)

A PLATO terminal has two character fonts: the standard font and the loadable font. (These are further subdivided into character memories, but that is an implementation detail.)

Each character is 16 pixels high by 8 wide. The standard font is fixed and unchangeable. It should be designed with

the target hardware specifically in mind for the best display because this character set is what users will see most frequently. In a machine with 256v by 512h resolution an 8 by 8 character grid would be required.

The loadable font (also known as the alternate character set) is transmitted to the terminal from the host or loaded from Micro PLATO as bit maps of the characters. The terminal resident must interpret these bit maps according to the needs of the hardware. A character might be transformed from 16 by 8 to 12 by 8 in a machine with 384v by 512h screen resolution.

This transformation will result in lost information when the target character size is smaller than the model's. One method when the target size is 8 by 8 is to logically or lines one and two, three and four, etc., to form an 8 by 8 character from the 16 by 8 bit image.

3.1.1.3. Keyboard (11/07/83)

The PLATO keyboard (often referred to as 'keyset') contains the standard set of alphanumeric keys, plus a set of specially named function keys, and another set of special mathematical and programming symbols. These keys all generate standard seven-bit ASCII character codes, but the host assigns them different interpretations from the standard set.

One difference between the PLATO keyboard and many others is that any key can be shifted to produce a different code, including function keys such as ENTER (alias RETURN, and called NEXT on PLATO). Some keyboards may not be able to detect such shifted function keys. In that case, those keys must be assigned to other keys on the keyboard, or assigned a multiple-key combination (shift-NEXT might be entered as ESC RETURN, and turned into the appropriate single character by the resident).

The standard PLATO keyset does not have certain characters explicitly labeled on it (such as @, #, etc.). These characters are generated on PLATO by multiple keypresses. Other terminals, however, do have these keys and must generate the appropriate two-character sequence when the user presses the key.

The PLATO to ASCII keycode mapping is shown in Appendix D.

3.1.1.4. Auxiliary input devices (11/01/83)

The standard PLATO terminal has an auxiliary input device used for pointing at objects on the screen. On Control Data terminals, this is a clear plastic touch panel placed over the display. The touchpanel is not active all the time. It is turned on only when the appropriate command is received from the host.

The touch panel divides the screen into a 16 by 16 square grid. Each square therefore covers a 32 by 32 pixel square area of the screen. When communicating these touch panel coordinates upline, the lower lefthand corner is (0,0) and

the upper righthand corner is (15,15).

The touchpanel can be simulated in a number of ways. A mouse, a light pen or even a joystick linked to a cursor could be used as a replacement.

3.1.1.5. Other I/O devices (11/01/83)

There are other devices (most notably serial I/O devices) that can be connected to PLATO terminals. Commands are available from the PLATO host to output data to and receive data from these devices. The Micro PLATO system can also access these devices through the calls specified in section 4.2.

3.1.1.6. Color (11/21/83)

The protocol supports the specification of foreground and background colors for display. Colors values are 24 bits in length, with 8 bit intensities for red, green and blue. The PLATO host will not send color commands to terminals which it does not think have color.

When one color is written on top of another color, the old color is replaced by the new color: they are not additive. This will be described in further detail in the sections on screen modes (see 3.2.3.1.1).

3.1.1.7. Screen printing (11/01/83)

The standard PLATO resident has the ability to make a copy of the screen on a printer in graphics mode. The screen copy can be initiated by a keypress or under program control (see section 4.2.26).

3.1.2. Resident modes and data formats (11/30/83)

The word 'mode' is used in three different ways when discussing the resident's operation.

The highest level is the PLATO versus TTY mode. In PLATO mode graphics commands are received and interpreted. In TTY mode only very simple functions are performed. Therefore a PLATO resident must actually be two programs in one: the first is a 'dumb TTY' emulator, which is used to get the user into the PLATO host; the second is the PLATO resident proper.

The intermediate level of 'mode' is how the resident interprets the data it receives (data mode). For example, the resident will be in text (character generation) mode when displaying text, line mode when drawing lines, block mode when drawing filled boxes, etc. There are eight or more of these modes, depending on how one counts.

The lowest level mode indicates how data should be displayed on the screen (screen mode). In text mode, for example, the resident can be in 'mode write', where characters are basically ored into screen memory. Or the resident can be

in mode inverse, which causes text to be displayed black on white instead of white on black (i.e., the background and foreground colors are reversed). There are four of these modes: rewrite, write, erase and inverse. In some data modes these four screen modes become only two. This will be explained in greater detail in the section on data modes.

Micro PLATO is not concerned with the data mode because it calls the graphics routines directly. Micro PLATO will never be in TTY mode, but it must operate while the host communications portion of the resident is in TTY mode. Micro PLATO uses the same definitions for screen modes, though the screen mode for Micro PLATO is set by a subroutine call.

3.1.2.1. TTY and PLATO modes (11/01/83)

The TTY mode need not be complicated. A bare minimum is that it operate in half-duplex (that is, echo what the user types on the screen) and display what it receives on the screen. It must also be able to recognize the escape sequence to select PLATO mode.

The PLATO mode of the resident is much more sophisticated. It receives the data from the PLATO host and displays the corresponding graphics and text.

There are escape sequences (detailed later) which switch between the PLATO and TTY modes of the resident.

3.1.2.2. Data modes (10/18/83)

When the terminal is in PLATO mode, there are a number of data modes which are used to transmit the graphics primitives: text, lines, points, blocks, character sets, raw data and data for programs that are running concurrently with the resident (in particular, Micro PLATO).

The mode is selected by means of a control character or an escape sequence. The terminal must remain in that mode until it receives another mode command.

3.1.2.3. Commands (10/18/83)

The terminal can at any time receive a command which should be executed immediately. Commands do not affect the current mode (except for commands such as the revert to TTY mode command).

For example, when in text mode, the resident can receive characters which are plotted on the screen, then receive a Select Size 2 command, which causes subsequent text to be displayed twice as large, then receive the Select Size 1 command, which would return character size to normal.

3.1.2.4. Transmission formats (11/07/83)

Data modes and commands require data to be in certain formats. These formats consist of one or more bytes with

their interpretation depending on what the particular mode or command needs.

All data characters are greater than 32 decimal (20 hex) to avoid confusion with control characters.

There are three basic formats used to transmit data to the terminal: bytes, words and coordinates. Some commands have data in formats that do not fall into any of these groups. These will be dealt with in the individual command descriptions.

(Note: Bits are numbered right to left from one. That is, the least significant bit is numbered one.)

3.1.2.4.1. Bytes (11/18/83)

Bytes are used in text mode to display characters. Each character received is plotted on the screen after being mapped through the currently selected character set memory.

3.1.2.4.2. Words (11/07/83)

Words are used to transmit logical quantities greater than eight bits in length. For example, when loading character sets, sixteen bits at a time are transmitted in character set load mode. Each word is encoded in three eight bit bytes. The format of these three bytes is:

```
   3rd Character        2nd Character        1st character
     received             received             received
   P 1 X X X X X X    P 1 X X X X X X    P 1 X X X X X X
       b18-------b13        b12-------b7         b6--------b1
```

Each byte contains six bits of data in the lower part of the byte. The next bit of the byte is always a 1. This ensures that data will not be interpreted as a command. The high bit is the parity bit, and is not relevant at this point.

For example, an echo command (see section 3.2.3.1.4) to query the terminal's subtype requires a word, of which seven bits are used. The value is 71 hex. The entire command would be

ESC Y q A @ (1B 59 71 41 40 hex).

3.1.2.4.3. Coordinates (11/18/83)

Coordinates are used to specify positions on the screen. They can be from one to four bytes in length. The point, line and block modes, as well as the load coordinate command, accept data in coordinate format.

The X and Y coordinates are converted to a sequence of four ASCII characters according to the following table:

	bit 7	bit 6	bits 5 through 1

high Y	0	1	5 MSB of Y coordinate
low Y	1	1	5 LSB of Y coordinate
high X	0	1	5 MSB of X coordinate
low X	1	0	5 LSB of X coordinate

(MSB = Most significant bits. LSB = least significant bits. Note that the coordinate is actually 10 bits in width. The top bit is always zero, always giving screen positions in the range 0 to 511.)

To save transmission time, not all the bytes are sent if certain requirements are met. The omitted bytes must be 'understood' by the terminal to be the same as the bytes of the last coordinate. Since bits 6 and 7 uniquely identify the low Y and the low X, and the high Y and high X can be distinguished from each other by context, all four bytes of the coordinate will not be transmitted if a portion of the last coordinate matches the new coordinate. For example, if the previous coordinate is (0,0), and the next coordinate is (10,0), only one byte (the byte describing the 5 lower bits of the new X coordinate) will be transmitted. If the coordinate following that is (30,30), only two bytes (for low Y and low X) will be sent.

PLATO terminal resident specification

The following table shows the coordinate byte transmission requirements.

COORDINATE BYTE TRANSMISSION REQUIREMENTS

BYTES WHICH CHANGE				BYTE TRANSMISSION REQUIRED			
High Y	Low Y	High X	Low X	High Y	Low Y	High X	Low X
			X				X
		X			X	X	X
	X				X		X
X				X			X
		X	X		X	X	X
	X		X		X		X
X			X	X			X
	X	X			X	X	X
X		X		X	X	X	X
X	X			X	X		X
	X	X	X		X	X	X
X		X	X	X	X	X	X
X	X		X	X	X		X
X	X	X		X	X	X	X
X	X	X	X	X	X	X	X
Sending Initial address				X	X	X	X
Returning to remembered address							X

The general rules are: 1) the order of the bytes is high Y, low Y, high X and low X (the low X coordinate is always sent last to indicate the termination of the coordinate); and 2), if the high X coordinate is changed, at least one of the Y coordinates is sent (so that the high X can be distinguished from the high Y).

For example, a load coordinate command (section 3.2.3.1.3) takes a coordinate as data. To set the (X,Y) position to (0,0), the sequence

ESC 2 SP #q SP @ (1B 32 20 60 20 40 hex)

would be sent. To move to (10,0) thereafter would be:

ESC 2 * (1B 32 2A hex)

and to move to (30,30) after that would be:

ESC 2 #n #x (1B 32 7E 5E hex)

3.1.3. Current (X,Y) screen location (11/18/83)

The current (X,Y) location on the screen is an important piece of information, used heavily by the resident. After a graphics object is put on the screen, the (X,Y) location is updated. The next object plotted may use this location as the starting point, updating the current (X,Y) location in turn.

3.2. Resident-host communications protocol (10/19/83)

The following sections describe in detail the PLATO ASCII communications protocol.

3.2.1. Transmission characteristics (11/18/83)

All characters sent to and received from the PLATO host consist of seven data bits, plus a bit of even parity. The parity on data received from the PLATO host can be ignored, but parity must be set correctly on data sent to the PLATO host from the terminal, because data with bad parity will be discarded.

One stop bit is sufficient.

The resident should be able to handle data from the host at 1200 bits per second, or approximately 120 characters per second. It is recommended that the receive communications channel be interrupt driven to ensure that no overrun occurs when the resident is engaged in lengthy operations or when Micro PLATO is in control. The resident should also accommodate 300 BPS transmission, and higher rates as desired (direct connection to the PLATO host can offer rates up to 9600 BPS).

The PLATO system runs in full duplex. That is, the host, rather than the modem or the terminal, echos all characters to the screen. or the terminal.

3.2.2. TTY mode requirements (10/24/83)

The TTY mode of the resident must be able to respond to the standard ASCII character set and control set. It should operate in half duplex (the network connecting the terminal to the PLATO host runs in half duplex).

The TTY mode must be able to recognize at least one escape sequence: ESC STX. This is the command from the PLATO host to the terminal to enter PLATO mode.

3.2.3. PLATO mode requirements (10/19/83)

3.2.3.1. Command formats (10/19/83)

Control characters and escape sequences are used to control the terminal's mode (text, point, line, etc.) and to

issue commands (such as erase screen, set writing mode, and so forth). The following sections describe the commands and modes and detail the control codes and escape sequences to use them.

Appendix B is a summary of the PLATO operation control codes and escape sequences.

3.2.3.1.1. Screen modes (10/20/83)

There are four 'screen modes' that control how graphics are written to the screen. They are called mode write, mode erase, mode rewrite and mode inverse. All four screen modes are available in text mode. In the other data modes, mode rewrite reduces to mode write and mode inverse reduces to mode erase (i.e., a line drawn in mode inverse will be black on a monochrome display).

3.2.3.1.1.1. Mode write (11/30/83)

Mode write is the screen mode that is most used by the PLATO system. It is useful to think of mode write as being an 'or' function, where the object being drawn on the screen (be it a character, line, point, or box) is merged with what is already present.

With solid graphic objects (lines, points and blocks), the object is simply put where it belongs. Characters are not so simple. Wherever there is an 'on' pixel in the character's bit map the corresponding pixel in the screen should be turned on. With a monochrome display this can be accomplished with an 'or', but with a color display it must be interpreted differently. For color, wherever there is an 'on' pixel in the character, the corresponding pixel must be set to the foreground color. Any 'off' pixels in the character must be left alone in the corresponding screen pixel.

The escape sequence to set the screen mode to mode write is ESC DC2 (1B 12 hex).

3.2.3.1.1.2. Mode erase (11/30/83)

Mode erase can be considered to be the opposite of mode write. For the solid graphic objects, pixels are turned 'off'. For characters, the screen pixels corresponding to the 'on' pixels in the character are turned 'off'. This is a complement and 'and' operation for monochrome displays: screen_pixel = screen_pixel AND NOT(character_pixel).

For color it means that the screen pixels corresponding to the 'on' character pixels must be set to the background color. Any 'off' pixels must be left alone.

The escape sequence to set mode erase is: ESC DC3 (1B 13 hex).

3.2.3.1.1.3. Mode rewrite (11/30/83)

Mode rewrite is akin to what is found on most 'dumb' TTY terminals. For graphic objects mode rewrite is the same as mode write, but for text it is much different. It is a straight assignment operation. In monochrome, the character is placed directly into screen memory, with on pixels turned on and off pixels turned off. This writes over (rewrites) whatever was previously displayed in the character area.

In color, the 'on' pixels are set to foreground color and the 'off' pixels are set to background.

The escape sequence to set mode rewrite is ESC DC4 (1B 14 hex).

3.2.3.1.1.4. Mode inverse (10/24/83)

Mode inverse is the 'opposite' of mode rewrite. It is also an assignment, but the character is complemented before being written into screen memory. In monochrome, every 'on' pixel in the character is turned 'off' on the screen and every 'off' pixel in the character is turned 'on'. For solid graphic objects mode inverse is the same as mode erase.

In color every 'on' character pixel is set to the background color and every 'off' character pixel is set to the foreground color.

The escape sequence to set mode inverse is ESC DC1 (1B 11 hex).

3.2.3.1.2. Data modes (10/24/83)

There are a number of modes in which the terminal processes data received from the host. These modes are selected by control characters or escape sequences. Data sent by the PLATO host is interpreted by each mode according to the data formats described in section 3.1.2.4.

3.2.3.1.2.1. Point mode (10/24/83)

The PLATO host sends an FS code (1C hex) to the terminal to put it in point mode (also known as mode 0). In this mode screen coordinates are sent to the terminal. These coordinates are in the format described in section 3.1.2.4.3. With a monochrome display device, the pixel is turned on in modes write and rewrite and turned off in modes erase and inverse.

With a color display the pixel is set to the foreground color in modes write and rewrite. It is set to the background color in modes erase and inverse.

The current (X,Y) position is left at this location.

3.2.3.1.2.2. Line mode (10/24/83)

In line mode line segments are drawn on the screen. Line

mode (also known as mode 1) is entered by receiving a GS (1D hex) code. In this mode, coordinates are sent to the terminal. The first coordinate received after transition to line mode only sets the first coordinate of the first line (no line is drawn). Thereafter, subsequent coordinates cause a line to be drawn from the previous coordinate to the current coordinate. The (X,Y) coordinate is left at its latest position when line mode is exited.

In mode write and mode rewrite, pixels are turned 'on' with monochrome and set to the foreground color with color. In modes erase and inverse pixels are turned 'off' with monochrome and set to the background color with color.

3.2.3.1.2.3. Load memory mode (11/22/83)

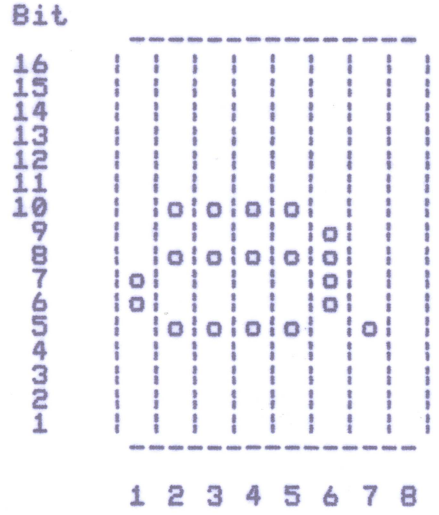
Load memory mode (also known as mode 2) is used for two different purposes. One is to load character sets into the terminal's programmable font, and the other is to load either executable code or binary data into terminal memory. When loading character sets a conversion is usually required to reformat the character into a usable form. Many terminals will not use the raw data mode for loading memory because they cannot execute the Z80 instruction set.

Before (and during) this mode, the PLATO host will send load memory address commands (see section 3.2.3.1.5) to specify where in memory the data should be loaded. The base which most terminals will be assigned is 3800 hex. (The Control Data IST III terminal with the short ASCII resident has its base located at 2340 hex.) The PLATO host will assume that the terminal's alternate character set memories start at 3800 hex. The resident must adjust the addresses sent down in the range 3800 to 4000 hex for storage into the character set memories. (See section 3.2.3.1.2.4 for a description of the character memories.)

In load memory mode data is received in 16-bit words (see section 3.1.2.4.2 for a description). The resident must maintain a variable which keeps track of the current load memory address, set by the load memory address command. This address must be incremented by two for each word received.

Load character set mode is initiated by the escape sequence ESC P (1B 50 hex). Each character consists of eight words of data. Each word is a vertical strip of the 16 high by 8 wide character. Bit one is at the bottom and bit 16 is at the top when the character is displayed on the screen. Many terminals' screen memories are not mapped to accommodate display of characters in this format, so it may be desirable to reformat the characters so that they are 16 bytes of eight-bit horizontal strips. The reformatting should take place after each character is received (i.e., after every eight words).

The following is an example of a character and the data words that would be transmitted from PLATO to form it:



The eight words for this character are (in hex):

0060
0290
0290
0290
01E0
0010
0000

The programmable character sets are not always completely utilized. If there are gaps in the character set, new load memory address commands will be sent to skip over the holes. The resident must account for this when receiving character sets.

The other memory load mode is initiated with the escape sequence ESC S (1B 53 hex). In this mode words are written into terminal memory at the address specified by the current load memory address. The low order byte (bits one to eight) is stored first, then the high order byte (bits 9 to 16) is stored in the following byte in memory. The load memory address is then incremented by two. Data received in this mode must not be converted.

Non-CDC terminals should ignore all data received in this mode.

3.2.3.1.2.4. Text mode (10/24/83)

The PLATO host puts the terminal in text mode (mode 3) with a US (1F hex) control character. In this data mode characters and text positioning commands are used to display text.

3.2.3.1.2.4.1. Character memories (11/30/83)

There are four character memories, known as M0, M1, M2 and M3. There are provisions built into the protocol to accommodate up to eight character memories, but only these four are currently in use. M0 and M1 are constant. M2 and M3 are the programmable font.

M0 contains the standard 96-character ASCII set.

M1 contains the PLATO extended character set. These are the special characters that PLATO uses which are not available in the standard ASCII set. The characters are illustrated in appendix A.

M2 is the first half of the programmable font. Character slots zero through 63 are placed in this character memory. The character set is loaded in the load memory mode (see section 3.2.3.1.2.3).

M3 is the second half of the programmable font. Character slots 64 through 127 are placed in this character memory.

There are eight commands to select the current character memory. The resident must start out with M0 as the default. When the PLATO host requires characters from the other character memories it will send down the appropriate command. The sequences for the commands are:

ESC B (1B 42): select M0
ESC C (1B 43): select M1
ESC D (1B 44): select M2
ESC E (1B 45): select M3
ESC F (1B 46): select M4
ESC G (1B 47): select M5
ESC H (1B 48): select M6
ESC I (1B 49): select M7

When a data character is received in text mode it is used as an index into the current character memory, and this bit map is put onto the screen in the current screen mode at the current (X,Y) location.

3.2.3.1.2.4.2. Character sizing commands (11/18/83)

Two character sizes should be available: size 0 (normal) and size 2 (bold). Size 2 characters are simply normal characters doubled in size, and are therefore 32 high by 16 wide. Size 2 also causes all character (and control functions) to perform on this greater scale. That is, after a size 2 character is plotted, the current X position will be increased by 16 rather than by 8.

Size 0 is the default size when the resident starts up.

Any other sizing of characters is done by the PLATO host with line drawings.

The sequence to select size 0 is ESC N (1B 4E hex). The size 2 sequence is ESC O (1B 4F).

3.2.3.1.2.4.3. Text direction commands (11/30/83)

Text can be plotted in four directions, controlled by text direction commands. The horizontal or vertical axis and the direction along that axis are selected with these commands. They control how the screen position is set after a character is plotted or a control function executed.

Forward plotting is selected by ESC L (1B 4C hex). This is the normal (and default) setting for the direction. The size of the character along the current axis is added to the current position along that axis to produce the next screen position. The other coordinate is not changed. For example, a size 0 character plotted horizontally would increase the X coordinate by 8. A size 2 character plotted vertically would increase the Y coordinate by 16 (twice the normal character width).

Backward plotting is selected by ESC M (1B 4D hex). This causes the character size along the current axis to be subtracted from the current axis coordinate. Note that the order of the pixels of the character on the display is not changed: only the positioning on the axis is affected.

Horizontal plotting is selected by the ESC J (1B 4A) sequence. This is the normal state and the default when the resident starts.

Vertical plotting is selected by ESC K (1B 4B). This causes characters to be plotted either up (if forwards writing is selected) or down (if backwards writing is selected) the screen.

When the vertical axis is selected for plotting, characters are rotated counterclockwise by 90 degrees.

If the vertical screen resolution of the terminal is less than the horizontal, vertical writing will not work well because the characters will not be a full eight dots wide.

3.2.3.1.2.4.4. Text positioning commands (11/30/83)

Text positioning commands are either ASCII control codes or two-character escape sequences. They change the screen position depending on the state of:

* the current axis (horizontal/vertical)
* the current direction (forwards/backwards)
* the current text size (normal or bold)

None of these functions is destructive (i.e., performing a backspace will not destroy the character backed up to). If screen wraparound occurs (if a coordinate becomes greater than 511 or less than 0) the coordinate is truncated to nine bits. 512 would thus become 0 and -1 would become 511.

The table below gives all the possible combinations of the axis, direction and character size.

Backspace (BS, 08 hex)

This causes the terminal to back up one character width along the current axis. 'Backing up' when in backwards

writing means going forward.

Tab (TAB, 09 hex)

This causes the terminal to advance one character width along the current axis. This function is like plotting an empty character.

Line Feed (LF, 0A hex)

This function decreases the display position by one character height perpendicular to the current axis (i.e., goes to the next line.) On the horizontal axis this moves down one line. On the vertical axis this moves right one character height.

Vertical Tab (VT, 0B hex)

This function advances the display line position by one character height perpendicular to the current axis (i.e., goes to the previous line).

Form Feed (FF, 0C hex)

This function sets the screen position to the first character on the top line of the display. The screen is not erased.

Erase Screen (ESC FF, 1B 0C hex)

This erases the screen and sets the screen position as form feed does. When the screen is erased all dots must be turned 'off' in monochrome or set to the background color in color. The palette should be reset if the underlying color hardware is palette-based (see section 3.2.3.1.8).

Carriage Return (CR, 0D hex)

This goes to the next line and sets the position along the current axis to the current margin.

Superscript (ESC @, 1B 40)

This increases the the screen position along the perpendicular axis by five pixels.

Subscript (ESC A, 1B 41 hex)

This decreases the screen position along the perpendicular axis by five pixels.

Set Margin (ESC Z, 1B 5A hex)

This command sets the margin used with the CR control code. The margin is set to the current X coordinate if horizontal writing is selected, and the current Y coordinate if vertical writing is selected.

The following table is a summary of the actions of the control codes.

PLATO DISPLAY CONTROL FUNCTIONS

ASCII CODE	FUNCTION	SIZE 0 HORIZONTAL FWD	SIZE 0 HORIZONTAL BKW	SIZE 0 VERTICAL FWD	SIZE 0 VERTICAL BKW	SIZE 2 HORIZONTAL FWD	SIZE 2 HORIZONTAL BKW	SIZE 2 VERTICAL FWD	SIZE 2 VERTICAL BKW
BS	Backspace	X-8	X+8	Y-8	Y+8	X-16	X+16	Y-16	Y+16
HT	Tab	X+8	X-8	Y+8	Y-8	X+16	X-16	Y+16	Y-16
LF	Line Feed	Y-16	Y-16	X+16	X+16	Y-32	Y-32	X+32	X+32
VT	Vert. Tab	Y+16	Y+16	X-16	X-16	Y+32	Y+32	X-32	X-32
FF	Form Feed	X=0 X=504 Y=496	Y=496	Y=0 Y=504 X=15	X=15	X=0 X=496 Y=480	Y=480	Y=0 Y=496 X=31	X=31
CR	Carriage Return	X=MARGIN Y-16	Y-16	Y=MARGIN X+16	X+16	X=MARGIN Y-32	Y-32	Y=MARGIN X+32	X+32
ESC @	SUPER	Y+5	Y+5	X-5	X-5	Y+10	Y+10	X-10	X-10
ESC A	SUB	Y-5	Y-5	X+5	X+5	Y-10	Y-10	X+10	X+10
Normal character		X+8	X-8	Y+8	Y-8	X+16	X-16	Y+16	Y-16

Note: Operations on X and Y coordinates are modulo 512.

3.2.3.1.2.5. Block mode (11/30/83)

Block mode is selected with EM (19 hex). In block mode (also known as mode 4) the host sends down pairs of screen coordinates which indicate opposite end points of a rectangular area on the screen to be drawn. In monochrome mode write and mode rewrite cause all dots inclusive of the specified area to be turned on; modes erase and inverse cause the area to be turned off (erased). In color, modes write and rewrite cause the area to be set to the foreground color while modes erase and inverse set it to the background color.

Once in block mode, subsequent pairs of coordinates indicate areas to be drawn. The coordinate pairs are each one to four bytes in the format specified in section 3.1.2.4.3. No assumptions should be made on the order of the coordinates received (i.e., the upper left hand corner of the rectangle is not always sent first).

The current (X,Y) position after each block is drawn should be set to the upper left hand corner (call it (X1,Y1)) minus 15 dots in the vertical direction (that is, (X1,Y1-15)).

3.2.3.1.2.6. User program mode (11/30/83)

The user program mode is used to communicate with "user programs." The main such user program is the Micro PLATO interpreter. When the resident receives data from the host in this mode it should call the routine whose address is stored in the memory location for this user mode. The addresses for these routines are kept in the Micro PLATO variables MOD5A, MOD6A and MOD7A. If the interface to Micro PLATO is not implemented, data in this mode should be ignored.

Mode five is selected by ESC T (1B 54 hex), mode six by ESC U (1B 55 hex) and mode 7 by ESC V (1B 56 hex). The host sends data in word format in this mode (see section 3.1.2.4.2 for a description of word format). The entire eighteen bits of the word are passed to the user program.

If the address stored in MODx is zero, then this data should be ignored and the resident should simplyreturn.

If the user program address is non-zero, it is a pointer to the routine to be called. The data is passed to the user program in the C, D and E registers on the Z80. Bits 1 through 8 are passed in E, 9 through 16 in D and 17 through 18 in the lowest two bits of C. See appendix C for the equivalent registers on other microprocessors.

3.2.3.1.3. Load coordinate command (10/24/83)

This command loads the current (X,Y) coordinates from the coordinate received from the PLATO host. The escape sequence ESC 2 (1B 32 hex) requires one to four bytes as its argument (in the format specified for coordinates in section 3.1.2.4.3).

The current (X,Y) coordinates are used to position text, reset line coordinates, etc.

3.2.3.1.4. Echo command (11/07/83)

The Echo command asks the terminal for information or requests a function. This command is used to query the terminal for its type, subtype, load file, configuration, terminal ID and memory contents. Seven bits of data (encoded in a three-byte response whose format is detailed in section 3.2.3.3.2) are sent up to the host. The Echo command is also used to sound the terminal's alarm.

The Echo command consists of the escape sequence ESC Y followed by three bytes in word format (see section 3.1.2.4.2). The low seven bits of the word indicate the echo code. The valid echo codes are listed below. If an echo code is received that is not one of those specified here, an echo response must be generated with that seven bit code in it.

Many of the echo codes are taken from variables used to communicate with Micro PLATO. The variable will be named here if this is the case. All such variables are described in section 4.3.

3.2.3.1.4.1. Terminal type (11/30/83)

Echo code 70 hex: Terminal type. This is 12 decimal for all ASCII terminals. This is the Micro PLATO variable M.TYPE. The resident must set this variable at initialization time.

3.2.3.1.4.2. Terminal subtype (11/30/83)

Echo code 71 hex: Terminal subtype. A different subtype is

assigned to each kind of machine. The following is a list of current subtypes:

0: Control Data IST II
1: Control Data IST III
2: Control Data 721-30 (Viking)
3: Control Data IST I
4: IBM PC
5: Zenith Z100
6: Apple II
7: Scrolling terminal (dumb TTY)
8: Dataspeed 40
9: VT 100 or ANSI X 3.64 compatible terminal
10: Unused
11: CDC 750
12: Tektronix 401x
13: TI 99/4a
14: Atari

As a general rule of thumb, if there is no subtype assigned to a particular machine the closest machine type should be used until one is assigned.

The PLATO host will send not send certain commands to terminal which cannot process them. For example, only terminals which can handle color will be sent color commands.

During terminal resident initialization, the Micro PLATO variable M.SBTYPE should also be set to this value.

3.2.3.1.4.3. Resident load file (11/03/83)

Echo code 72 hex: Terminal resident load file. This is only applicable to Control Data terminals whose residents are loaded from the PLATO host. The standard load files are given below:

0: ASCII/Graphics resident (IST III)
1: Short ASCII resident (IST II and III)
2: Reserved
3: IST I ASCII resident
4: Reserved
5: Reserved
6: Reserved
7: Reserved
8: Viking 721-30 resident
9: Reserved

Other terminals should return 0 as their load file. The Micro PLATO variable M.LDFILE is also set to this value.

3.2.3.1.4.4. Terminal configuration (11/03/83)

Echo code 73 hex: Terminal configuration. This indicates the presence or absence of certain terminal features. Currently only two bits are defined:

Bits 1 - bit 5: undefined (should be 0)
Bit 6: = 0, 16K memory; = 1, 32K memory
(only applicable to CDC ISTs,

should be 0 for non-CDC terminals.)
Bit 7: = 0, no touch panel; = 1, touch panel (or other device) present

The Micro PLATO variable M.CONFIG is also set to this value.

3.2.3.1.4.5. Send backout code (11/03/83)

Echo code 7A hex: Send a backout code to PLATO host and turn turn off the Data Terminal Ready (DTR) signal. This is for disconnecting with the host. It is infrequently used.

3.2.3.1.4.6. Sound alarm (11/03/83)

Echo code 7B hex: Sound audible alarm at the terminal. Nothing should be sent back to the host.

3.2.3.1.4.7. Return terminal ID code (11/03/83)

Echo code 7C hex: Return 16-bit terminal ID code. (See section 3.2.3.3.2.) Only CDC terminals need respond to this. Others may ignore this code and return a normal echo response.

3.2.3.1.4.8. Read memory (11/07/83)

Echo code 7D hex: Return the 7 least significant bits of the contents of the memory location specified by a preceding Load memory address command. Non-CDC terminals may ignore this echo code and echo a normal response.

3.2.3.1.5. Load memory address command (11/03/83)

The load memory address command consists of the escape sequence ESC W and three bytes formatted as word data (see section 3.1.2.4.2). The lower sixteen bits of the word form the absolute memory address where data should be written. The resident must maintain a variable with the current memory address, which is used by other commands and modes.

As described in section 3.2.3.1.2.3, Load memory mode, this is used to set the addresses for loading character sets. Non-CDC terminals should interpret these addresses as indices into a byte array for the programmable character set, offset by 3800 hex.

3.2.3.1.6. SSF command (10/24/83)

The SSF (Select Special Function) command is used by the PLATO host to communicate with special devices attached to the terminal. The command consists of the sequence ESC Q (1B 51 hex) followed by three bytes formatted as a word (see section 3.1.2.4.2). Only the lower fifteen bits of the word are used.

Bits 15 through 11 are the device address. These must be examined to determine what kind of SSF command this is.

3.2.3.1.6.1. Select slide projector (10/24/83)

This SSF command was used to control the operation of the image projector on an obsolete plasma terminal and is not used on modern terminals. It should therefore be discarded.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X

Note that the device address bits (15 to 11) are 00000.

3.2.3.1.6.2. Set interrupt mask (10/24/83)

This SSF command controls the interrupt mask. Its primary use is to enable and disable the touch panel, but it also controls the interrupt of the serial channel. Device address bits are 00001, and bits 1 thru 8 are the terminal interrupt mask. If a bit position is a one, the corresponding device is enabled. If the bit is a zero, that device is disabled. Currently only two devices are controlled with this: the touch panel (or replacement device) and the serial channels.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	0	0	0	0	1	X	X	<---INTERRUPT MASK--->							

bit 1 = not used
bit 2 = not used
bit 3 = not used
bit 4 = serial channels
bit 5 = always 1
bit 6 = touch panel
bit 7 = always 1
bit 8 = always 1

3.2.3.1.6.3. Select input/output (11/30/83)

This instruction selects a device address for further transfers with the EXT command (see section 3.2.3.1.7). In addition, it may be used to transfer one eight-bit data byte to the device. The bit definitions are as follows:

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	A	A	A	A	A	W	I	D	D	D	D	D	D	D	D

Bits 15 to 11 Address -- Selects the device that is to communicate with the terminal. Device address must be two or greater.

Bit 10 Read/Write -- If bit 10 is one, data is read from the device and sent upline to the host according to the format in section 3.2.3.3, IF bit 9 is zero.
If bit 10 is zero, data in bits 1 through 8 are sent to the device, IF bit 9 is zero.

Bit 09 — Inhibit -- When this bit is one, all data transfers are inhibited. This is used to set a device address for subsequent EXT commands without performing any data transfer. When this bit is zero the transfer indicated by bit 10 takes place.

Bits 08 to 01 — Data -- This data is transferred to the device if both bits 09 and 10 are zero.

The resident should retain the input and output device addresses separately. The addresses may be used by subsequent EXT commands.

Device addresses 00 and 01 are reserved for the 'Slide Projector' and 'Set Interrupt Mask' formats of the SSF instruction.

Device addresses 02 and 03 are used to transfer data and control to serial port devices. Writing to address 02 will transfer a byte of data to the currently selected serial device, reading from address 02 will transfer a byte from the device. Writing to address 03 will set up control information for the selected serial device. The codes used for control are specified in appendix F. Reading from address 03 will give the current status, also defined in appendix F.

Device address 04 is used to select a specific serial port for terminals with multiple serial ports. Writing to device 04 selects the device. Reading from address 04 should return the currently selected device number. The mapping of these logical devices to physical devices is up to the implementor, but the following conventions have been established:

Port 0: the default serial device.
Port 1: other serial device.
Port 2: other serial device.
Port 3: other serial device.
Port 4: other serial device.
Port 5: other serial device.
Port 6: other serial device.
Port 7: Internal modem on Viking 721-30
Port 8: System printer.
Port 9: Communications channel to the PLATO host.

Device 05 is not used.

Addresses 06 through 31 (decimal) are devices on the parallel channel of the CDC terminals.

3.2.3.1.7. EXT (External Data) command (10/24/83)

This command transfers two 8-bit bytes to the external device selected by the previous SSF command.

The EXT command consists of the sequence ESC R (1B 52 hex) followed by three bytes in word format (see section 3.1.2.4.2). The lower sixteen bits of the word are used: first, bits 9 through 16 are transferred to the selected device as one byte; then bits 1 through 8 are transferred.

3.2.3.1.8. Color commands (11/18/83)

There are two commands for selecting color: ESC a (1B 61 hex) sets the foreground color. ESC b (1B 62) sets the background color. Color values are 24 bits in length, 8 bits of intensity for each of the colors red, green and blue, in the following format:

```
bits   24 through 17     16 through 9     8 through 1
       <----red---->     <--green--->     <--blue--->
```

These 24 bits are placed in the four bytes following the first two bytes of the command accordingly:

```
                b8 b7 b6 b5 b4 b3 b2 b1
Byte 1:          P  1 <-bits 01 to 06->
Byte 2:          P  1 <-bits 07 to 12->
Byte 3:          P  1 <-bits 13 to 18->
Byte 4:          P  1 <-bits 19 to 24->
```

Notes: Bit 8 (P) is the parity bit (even) and bit 7 is always 1 to avoid sending control codes.

The PLATO software can specify any color in this range, but eight colors have predefined values in the PLATO Author Language. These are:

Color	Hex value
black	000000
blue	0000FF
green	00FF00
cyan	00FFFF
red	FF0000
magenta	FF00FF
yellow	FFFF00
white	FFFFFF

If a color is received and is not available on a particular machine, the closest possible match should be used.

Many display devices use a palette scheme to select colors. The resident must then keep track of which colors are selected. When a new color is received, the next free slot in the palette should be assigned to this color (if a color is not available, the closest match should be used). Subsequent settings to this color will cause no additions to the table. The color will simply be reused. When the palette table becomes full, further color commands should be ignored.

Whenever the screen is erased, the palette table should be emptied and the current background and foreground colors placed in the table.

3.2.3.1.9. Paint command (11/18/83)

The paint command is used to fill an arbitrary area with a certain color or character pattern. The escape sequence for the paint command is ESC c (1B 63), which is followed by two bytes which contain a nine-bit value formatted in this way:

```
bits          b9 b8 b7 b6 b5 b4 b3 b2 b1
               M  M  C  C  C  C  C  C  C
```

Bits 8 to 9: The character set memory used for the character. 00 is M0, 01 is M1, etc.
Bits 1 to 7: An index into the character set memory specified in bits 8 and 9.

This information is placed in the data bytes following the escape sequence this way:

```
                b8 b7 b6 b5 b4 b3 b2 b1
Byte 1           P  1 <--bits 1 to 6-->
Byte 2           P  1  0  0  0 <7 to 9>
```

Note: b8 is the parity bit.

If the value of the character is zero, then a solid paint is done over the entire area. The paint begins at the current (X,Y) position (set perhaps by a load coordinate command). Pixels surrounding the start location are set to the foreground color until a pixel set to the background color or the edge of the screen is reached.

When filling with a character the character pattern is repeated throughout the same area that a solid fill would have covered. For reasons of speed, the characters should be aligned on byte boundaries in the filled area. The character is written into memory in mode write.

Since the paint operation can take a very long time, there must be a check in it for the user aborting it. A check should be made for the STOP or the shift-STOP keys in the outer loop of the paint. Other keys should be processed as normal (sent to the host or buffered for Micro PLATO as required). The STOP or SHIFT-STOP key must also be sent if pressed during a paint.

3.2.3.1.10. Resident mode change commands (11/30/83)

The PLATO, TTY and 'graphics' modes of resident operation are selected by three escape sequences. ESC STX (1B 02 hex) selects PLATO operation. The host will always send this command to the terminal at the beginning of the session.

ESC ETX (1B 03 hex) selects the TTY mode of operation. At the end of the PLATO session the host will send this sequence to the terminal to reset to TTY.

ESC SOH (1B 01 hex) is used to select the 'graphics' mode in the Viking 721-30 and IST III graphics residents. In this mode the terminal emulates a Tektronix 401x graphics terminal. Other machines should treat this as a command to enter TTY mode.

3.2.3.3. Upline data formats (11/30/83)

The PLATO resident sends a number of different types of data to the host. Besides keys the user presses, the resident

sends echo responses, touch input, external device data and unsolicited status. All upline data are referred to as 'keys.'

The PLATO host considers keys to be ten bits in length: the bottom eight bits are the data and the upper two bits are an identifier. Since only seven bits of data can be sent to the host at a time, two formats are used for transmitting keys upline. The majority of keys are typed by the user, so these keys are sent as single characters to the host, while the non-keyboard, ten-bit data types are sent in three-character escape sequences. The general format for these sequences is:

bit	10	9	8	7	6	5	4	3	2	1
	I	I	D	D	D	D	D	D	D	D

Bits 10 and 9 — Identifier for this key type. The values for these bits are:

0 0: Echo response
0 1: Touch panel data
1 0: External data
1 1: Unsolicited status

Bits 8 to 1 — Data for this key.

These ten bits are formatted into the three characters sent to the PLATO host in the sequence ESC c1 c2, where c1 and c2 are:

First character after ESC (c1):

b8	b7	b6	b5	b4	b3	b2	b1
P	1	<----	bits	1	to	6	----->

Second character after ESC (c2)

b8	b7	b6	b5	b4	b3	b2	b1
P	1	1	0	<---	10	to	7-->

All characters sent upline must have even parity. Parity may be generated by hardware or by software (in which case the hardware would be set up to transmit 8 bits with no parity). The PLATO host will discard any data received with bad parity.

3.2.3.3.1. Keyboard data (11/18/83)

Keyboard keys are sent up to the PLATO host as single bytes. They are generally 7 bit quantities (with the top bit used for even parity). Appendix D gives the PLATO to ASCII key mapping.

Some keyboards have keys that are represented by a single character in the ASCII character set, but by a two-character sequence in the PLATO character set. The two characters are ACCESS (also known as SHIFT-SQUARE, a PLATO key for generating these and other special characters) and another character as specified in the following table.

Character	Hex value (ASCII)	Resulting sequence	
#	23	ACCESS $	(00 24)
&	26	ACCESS +	(00 2B)
@	40	ACCESS 5	(00 35)
\	5C	ACCESS /	(00 2F)
#x	5E	ACCESS x	(00 78)
#q	60	ACCESS q	(00 71)
{	7B	ACCESS [	(00 5B)
!	7C	ACCESS I	(00 49)
}	7D	ACCESS]	(00 5D)
#n	7E	ACCESS n	(00 6E)

Note that some of these characters (#x, #q, and #n) are auto-backspacing, auto-raising diacritics on PLATO. When typed, they will adjust themselves over the previous character in a single keypress (o#x, n#n and E#q, for example). This is a function of the PLATO system and nothing further is required in the terminal resident to support this.

3.2.3.3.2. Echo responses (10/31/83)

When an echo command is received (see section 3.2.3.1.4) the terminal will in most cases send back an echo response to the host. The logical ten-bit key is formatted as follows:

bit	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1
	0	0	1	<--seven bit response---->						

The ID code echo requires a four-byte response, which is formatted as follows (note: only CDC terminals send this):

	b8	b7	b6	b5	b4	b3	b2	b1
First byte	P	0	0	0	ID bits 04-01			
Second byte	P	0	0	1	ID bits 08-05			
Third byte	P	0	1	0	ID bits 10-09			
Fourth byte	P	0	1	1	ID bits 16-13			

The 16-bit terminal ID code is transmitted to the PLATO host in 4-bit segments, with the least significant bits sent first, (bits 01-04 of the first word). Bits 06 and 05 of the output word identify which 4 bits of the ID code are contained in the output word. Bit 7 is always 0.

3.2.3.3.3. Touch panel data (11/08/83)

The touch panel has a resolution of 16 by 16. Thus, a touch key can fit into eight bits. Its ten-bit form is:

Touch Panel Output Word	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1
	0	1	<--X coord--->				<--Y coord--->			

The lower left hand corner of the screen is at location (0,0) while the upper right hand corner is at (15,15). Alternate input devices that have greater precision must be scaled down to fit into the 0 to 15 range.

The touch panel should be active only when enabled by an SSF command (see section 3.2.3.1.6.2).

3.2.3.3.4. External data (11/01/83)

There are two cases where EXT data is sent to the host: 1) When an SSF command is received with bit 10=1, which requests an input from a device connected to the terminal. 2) If interrupts are enabled for the external devices and such a device interrupts. In either case, one byte of data should be read from the designated device and sent to the host in a ten-bit key which is formatted in this way:

	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1
External Data Output Word	1	0	<-------EXTERNAL DATA-------->							

3.2.3.3.5. Unsolicited status (11/18/83)

The unsolicited status word is transmitted to the PLATO host as a result of some event occurring at the terminal.

The following status codes describe bits 01 to 07 of the unsolicited status word. The description describes the event that initiated the sending of the unsolicited status to the PLATO host.

Status Code	Description
00	Unassigned
01	Unassigned
02	Terminal Master Reset
03	Unassigned
04	Unassigned
05	Mode 2 Block error (not used)
06 to 3F	Unassigned
40 to 7E	Reserved for use by PCN loader
7F	PLATO Backout Code

The status byte is placed in the ten-bit key as follows:

	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1
Unsolicited Status Output Word	1	1	1	<------STATUS CODE------->						

3.2.3.4. Redirecting I/O (11/30/83)

When the interface to Micro PLATO is implemented in the resident, input from the keyboard, touch panel and external devices must be redirected to Micro PLATO rather than sent to the host if Micro PLATO has requested this.

The variable M.KSW is set by the Micro PLATO interpreter if this input data is to be redirected to Micro PLATO.

When input is generated the contents of M.KSW must be examined, and if required the data must be placed in the buffer for Micro PLATO instead of sent to the host.

Interrupts from external devices (such as serial ports) must also check the M.EXTPA (external processor) address to determine whether Micro PLATO is requesting to handle the interrupt. If M.EXTPA is zero, a data byte is read in from

the interrupting device and is sent as an external key (see section 3.2.3.3.4 for the description of this). If the processor address is non-zero, this address should be called so that Micro PLATO can handle the interrupt. Note that M.EXTPA can be zero and M.KSW can indicate that the key go to Micro PLATO.

3.3. Exception handling (11/07/83)

Certain error conditions require special handling to restore the resident to a 'good' state. This section describes these conditions and the actions that should be taken.

3.3.1. Loss of carrier (11/07/83)

If the carrier is lost on the line to the host, the resident should go into TTY mode and notify the user of this. The resident should not stay in PLATO mode because it will frequently not be in text mode, and any messages sent by the supporting network would not be visible.

There should be a timeout on the carrier of approximately 1.5 seconds. If carrier has not been detected for this time, the resident should revert to TTY mode.

This check should not be made when Micro PLATO is being executed.

3.3.2. Buffer overflow (11/30/83)

At a minimum, the resident must be able to plot all output received at 1200 BPS without overflowing its input buffer. If overflow occurs, the resident must be able to recover and continue to function.

A method for flow control is in the planning stages.

3.3.3. Host crash (11/14/83)

If the PLATO host crashes without dropping carrier, there should be a way for the user to escape from PLATO mode, for the same reasons the resident should revert to TTY mode when the carrier is lost. On the IST III terminal this is accomplished by a pressing the reset button on the terminal for three seconds. On the Viking 721-30 pressing the keys Control-Shift-MREL performs this function.

3.4. Programming guidelines (11/30/83)

The terminal resident can be a multipurpose program. It can communicate solely with the PLATO host over the communications line, or it can load a Micro PLATO interpreter and execute Micro PLATO lessons, or it can do both simultaneously, facilitating communications between the host system and Micro PLATO lessons.

Currently Micro PLATO interpreters exist for the Z80 and 8086 families of microprocessors. Residents for machines

of this class should be structured such that the low level routines can be called by the routines that handle communications with the host, as well as the entry points listed in section 4 of this document (the interface to Micro PLATO).

There are a number of factors dealing with the way output from the host is transmitted that dictate how the resident should be written. For example, it is very common on PLATO for output to be aborted. This means that at any point in time an escape sequence or the data for a command could be truncated and another control character or escape sequence sent. Users abort output frequently when they press keys before the host is finished sending a page.

To avoid misinterpretation, the resident should be driven by the data, rather than waiting for each individual component of it. Each character from the host should be processed in the following manner:

```
IF the previous character was ESC THEN
     Perform the action required for this escape
     sequence.  This might involve setting a new data
     processing mode, or initiating a command that will
     wait for a single unit of data and then go back to
     the current data processing mode, or a simple
     command to set the screen mode.
     Note that it is possible to receive the sequence
     ESC ESC (if output is aborted).  This must be
     treated identically to receiving a single ESC.
ELSE IF this character is a control character THEN
     Perform the action required for this control
     character.  As above, this might select a new
     mode or cause simple repositioning of the current
     (X,Y) location (such as CR).
     When ESC is received it must be remembered so
     the first test above can be performed.
ELSE IF a command is in progress THEN
     handle this byte appropriately.  Commands that
     fall into this category include load coordinate,
     set memory address, SSF commands, etc.
ELSE
     Give this data to the appropriate routine handling
     the current mode (text, point, line, etc.)
ENDIF
```

Since so much of the data follows the same format (most commands and modes expect data in word or coordinate format) it is useful to write these two routines once. The argument to such a get_coordinate routine would be the character received from the host; the output would be an indication of whether the coordinate was completed and if so the coordinate received. One piece of information that all routines need is the number of bytes received since the last unit of data was completed. This byte count should be zeroed at the beginning of every command and setting of data processing mode. This is because output may be aborted at any time and a command or data word interrupted before it is completely sent.

4. Resident-Micro PLATO interface (11/18/83)

This section describes the interface between the resident and Micro PLATO. This is, of course, very hardware specific. The basic interface will be given in terms of the Z80 implementation. A mapping between this and other processors is given in Appendix C.

4.1. General overview (11/08/83)

Micro PLATO does not send escape sequences to the resident to perform terminal functions. Rather, the interpreter makes calls to the resident through a jump table (called the "R." table) and communicates through a set of variables (called the "M." variables).

When performing services for Micro PLATO the resident should use the M. variables for specifying the current screen mode, text plotting direction, etc., instead of the variables used for communicating with the host. The resident should not use the host variables (for current (X,Y) position, etc.) when performing functions for Micro PLATO because the host can be connected to the terminal at the same time Micro PLATO is running, sending output to the resident simultaneously with Micro PLATO.

The rules for loading the interpreter and the interface for a particular processor are given in Appendix C.

4.2. Callable routines (11/03/83)

Micro PLATO uses the following conventions when calling the resident routines.

1. Entry is made via a 'call' instruction to a fixed address in a jump table. The offsets for this jump table are dependant on the processor. A table is given for each microprocessor in Appendix C.

2. All data is passed to the resident routines in the following manner except where specified.

   a. One variable -- passed in HL register pair.
   b. Two variables -- passed in HL and DE register as listed for that routine.
   c. Three or more variables -- variables stored in memory, with HL = base address of variable string, and DE = length of variable string.

3. The resident may use and destroy any registers not explicitly listed as safe in the Appendix for each processor.

4. The user operating under modes 5, 6, or 7 will receive host data from the resident in C, D and E registers as shown in section 3.2.3.1.2.6.

5. Micro PLATO will restore the stack pointer to its original value when it exits. It will otherwise only push and pop data from it, as well as perform procedure calls.

4.2.1. R.INIT (11/18/83)

Micro PLATO will jump to this entry point when it is finished. There should be no return. The course to be followed thereafter will depend on the implementation. Terminals that are terminals only (CDC IST and Viking 721-30 terminals) should put themselves in TTY mode. Microcomputers emulating terminals may do likewise, or return to the operating system, or return to a page of options for the user to select what happens next.

4.2.2. R.DOT (11/08/83)

This is a point plot routine which will write or erase a single dot on the display. The coordinates for the dot are passed to the routine with X coordinate in HL register pair and Y coordinate in DE register pair. The screen mode is set in M.MODE prior to calling R.DOT.

At the completion of this routine, X and Y coordinates are set to the coordinates passed to the routine in HL and DE register pairs.

Entry: DE = Y coordinate
HL = X coordinate
Screen mode in M.MODE

Exit: (X,Y) set to this coordinate.

4.2.3. R.LINE (11/18/83)

This routine will plot a line from the current (X,Y) position on the display to the coordinates passed to the routine. The new X coordinate is passed in the HL register pair and Y in the DE register pair. The line is either erased or written depending on the condition of the screen mode in M.MODE. At the completion of the routine the current (X,Y) coordinate is updated to the coordinates passed.

Entry: DE = ending Y coordinate
HL = Ending X coordinate
Current (X,Y) set to starting coordinate
Screen mode set in M.MODE

Exit: (X,Y) set to the coordinates passed.

4.2.4. R.CHARS (11/30/83)

R.CHARS displays a string of six-bit PLATO character codes. They are displayed in the screen mode specified in M.MODE, with the plotting parameters specified by M.CCR. The address of the string is passed in the HL register pair. The conversion of PLATO character codes to the ASCII codes used in the resident is described in the next section. Control characters are indicated by two character sequences, the first of which is the 'uncover code.' The value of the uncover code is 3F hex. Control codes are listed in section 4.2.4.2.

The end of the string can be indicated in two ways, both of which use the uncover code followed by another byte. 3F 00 hex signifies the end of a normal string. 3F 01 hex signifies the end of the string and indicates that the uncover flag in M.MODE should be set. This allows an uncover code to be the last character in a string. On the next call to R.CHARS the uncover flag should be examined and if set, the first character in the string should be treated as control.

The following example illustrates the string required to perform a line feed, display the word PLATO, and perform a carriage return (assuming that M1 is the selected character memory).

3F	Line
0A	Feed
10	P
0C	L
01	A
14	T
0F	O
3F	Carriage
0D	Return
3F	End of String
00	Code.

Entry:
HL points to string of characters terminated by end of string control.
M.MODE controls screen mode and holds the uncover flag.
M.CCR controls character size, etc.
(X,Y) set to location where string will be plotted.

Exit:
(X,Y) updated to point after the last character.
The uncover flag in M.MODE is set if the last code in the string was 3F 01, otherwise it is cleared.

4.2.4.1. PLATO character code conversion (11/18/83)

Micro PLATO has a different conception of the character memories from the ASCII residents. These originate historically from the use of non-ASCII terminals with PLATO. Most notably, Micro PLATO places all lower case alpha characters in M0, while upper case alpha is in M1. In the resident all alpha characters are in M0. R.CHARS must translate these Micro PLATO codes into the ASCII characters that can be plotted. The character memory bits in M.CCR indicate the Micro PLATO character memory to be used. The tables below give the equivalent ASCII memory and character code.

Micro PLATO assumes that its M1 table follows directly after M0, so it will on occasion index into M0 with values greater than the number of characters in M0. These should be treated as indices into M1 (adjusted appropriately).

4.2.4.1.1. Micro PLATO M0 (11/03/83)

Micro PLATO M0 character memory

Micro hex	PLATO char	Resident mem	Resident hex	Micro hex	PLATO char	Resident mem	Resident hex
00	:	M0	3A	20	5	M0	35
01	a	M0	61	21	6	M0	36
02	b	M0	62	22	7	M0	37
03	c	M0	63	23	8	M0	38
04	d	M0	64	24	9	M0	39
05	e	M0	65	25	+	M0	2B
06	f	M0	66	26	-	M0	2D
07	g	M0	67	27	*	M0	2A
08	h	M0	68	28	/	M0	2F
09	i	M0	69	29	(	M0	28
0A	j	M0	6A	2A	)	M0	29
0B	k	M0	6B	2B	$	M0	24
0C	l	M0	6C	2C	=	M0	3D
0D	m	M0	6D	2D	SP	M0	20
0E	n	M0	6E	2E	,	M0	2C
0F	o	M0	6F	2F	.	M0	2E
10	p	M0	70	30	/	M1	2F
11	q	M0	71	31	[	M0	5B
12	r	M0	72	32	]	M0	5D
13	s	M0	73	33	%	M0	25
14	t	M0	74	34	*	M1	2A
15	u	M0	75	35	@	M1	24
16	v	M0	76	36	'	M0	27
17	w	M0	77	37	"	M0	22
18	x	M0	78	38	!	M0	21
19	y	M0	79	39	;	M0	3B
1A	z	M0	7A	3A	<	M0	3C
1B	0	M0	30	3B	>	M0	3E
1C	1	M0	31	3C	_	M0	5F
1D	2	M0	32	3D	?	M0	3F
1E	3	M0	33	3E	>	M1	3F
1F	4	M0	34	3F	uncover code		

4.2.4.1.2. Micro PLATO M1 (11/08/83)

Micro PLATO M1 character memory

Micro hex	PLATO char	Resident mem	Resident hex	Micro hex	PLATO char	Resident mem	Resident hex
00	#	M0	23	20	#W	M1	26
01	A	M0	41	21	->	M1	27
02	B	M0	42	22	#X	M1	28
03	C	M0	43	23	<-	M1	29
04	D	M0	44	24	~	M0	7E
05	E	M0	45	25	+	M1	2B
06	F	M0	46	26	-	M1	2C
07	G	M0	47	27	u	M1	2D
08	H	M0	48	28	^	M1	2E
09	I	M0	49	29	{	M0	7B
0A	J	M0	4A	2A	}	M0	7D
0B	K	M0	4B	2B	&	M1	26
0C	L	M0	4C	2C	<>	M1	25
0D	M	M0	4D	2D	SP	M1	20
0E	N	M0	4E	2E	*	M1	2A
0F	O	M0	4F	2F	#o	M1	3D
10	P	M0	50	30	=	M1	22
11	Q	M0	51	31	#a	M1	30
12	R	M0	52	32	#b	M1	31
13	S	M0	53	33	#d	M1	32
14	T	M0	54	34	#l	M1	33
15	U	M0	55	35	#m	M1	34
16	V	M0	56	36	#p	M1	35
17	W	M0	57	37	#r	M1	36
18	X	M0	58	38	#s	M1	37
19	Y	M0	59	39	#w	M1	38
1A	Z	M0	5A	3A	<=	M1	39
1B	#n	M1	23	3B	>=	M1	3A
1C	/	M1	21	3C	#t	M1	3B
1D	#x	M0	5E	3D	@	M0	40
1E	#e	M1	46	3E	\	M0	5C
1F	#q	M0	60	3F	uncover code		

4.2.4.1.3. Micro PLATO M2 and M3 (11/04/83)

The alternate font character memories are indexed directly by the Micro PLATO character code (this is where the arrangement of the character memories originated). The Micro PLATO interpreter will assume that these two memories are contiguous when referring to characters in the M3 memory by indexing into the M2 memory with an out-of-range character code, in the same way it treats M0 and M1 as contiguous.

4.2.4.2. PLATO control code conversion (11/18/83)

Control codes are formed by the uncover code followed by one byte. This byte is an ASCII control code or a value that can be transformed into the second character of an escape sequence from the host. The control codes are listed below.

Byte after uncover	Purpose
00	End of string

01	End of string, uncover flag set
08	Backspace
09	Tab
0A	Linefeed
0B	Vertical tab
0C	Form feed
0D	Carriage return

(The previous codes are identical to the control codes sent by the host. The following codes can be converted to the second character of the escape sequence by adding 32 hex.)

0E	Super
0F	Sub
10	Select M0
11	Select M1
12	Select M2
13	Select M3
14	Select M4
15	Select M5
16	Select M6
17	Select M7
18	Horizontal
19	Vertical
1A	Forward writing
1B	Reverse writing
1C	Select size 0 characters
1D	Select size 2 characters
3F	Uncover

These functions are affected by the values stored in the M. variables (M.MODE, M.MARGIN, M.CCR).

4.2.5. R.BLOCK (11/18/83)

R.BLOCK erases or fills an area of the screen according to M.MODE. The coordinates of two opposing corners of the area are placed in memory as described below. HL points to the address of the first byte of the list. Since any two opposing corners can be selected, the resident must not make any assumptions about which corner is which.

At the completion of the block operation, the screen location should be set to the upper left hand corner of the block, minus 15 in the direction (i.e., if (X1,Y1) is the upper left hand corner of the block, the current (X,Y) location should be set to (X1, Y1-15)).

Entry HL = Memory Address of Coordinates
M.MODE contains screen mode
Exit condition
X = X1 and Y = Y1 - 15

(HL) + 0 = X1 lower 8 bits
(HL) + 1 = X1 upper bit
(HL) + 2 = Y1 lower 8 bits
(HL) + 3 = Y1 upper bit
(HL) + 4 = X2 lower 8 bits
(HL) + 5 = X2 upper bit
(HL) + 6 = Y2 lower 8 bits
(HL) + 7 = Y2 upper bit

4.2.6. R.INPX (11/18/83)

This routine will read the current X coordinate and return to the caller with the coordinate in HL register pair. The resident should always truncate this to 9 bits.

Entry - No preset condition required.
Exit - HL = Current X coordinate.

4.2.7. R.INPY (04/12/83)

This routine will read the current Y coordinate and return to the caller with the coordinate in HL register pair.

Entry - No preset condition required.
Exit - HL = Current Y coordinate.

4.2.8. R.OUTX (11/18/83)

This routine sets the contents of HL register pair as the new X coordinate for the display. The Y coordinate is not affected. The resident must always truncate this value to nine bits (in the range 0 .. 511).

Entry - HL = New X coordinate.

4.2.9. R.OUTY (11/18/83)

Sets the current Y coordinate to the contents of HL. The X coordinate is not affected. The resident must always truncate this value to nine bits.

Entry - HL = New Y coordinate.

4.2.10. R.XMIT (11/03/83)

This routine provides communication from Micro PLATO to the host. HL contains a 10-bit PLATO keycode. The bottom eight bits are the key while the upper two bits are the key identifier. This identifier is basically the same as the identifier described in section 3.2.3.3, except that ID bits 00 are keyboard data rather than echo response data. Thus, for ID bits 01 through 11 the ten-bit key is transmitted to the host as described in section 3.2.3.3.

Keyboard data, however, must be converted from the internal Micro PLATO codes to ASCII before being sent. The table in Appendix D describes the required translation from Micro PLATO format to ASCII.

4.2.11. R.MODE (11/21/83)

This routine can be used to change operating modes of the terminal, set the screen mode for Micro PLATO, and erase the screen. This routine should be called to change M.MODE. If M.MODE is changed directly the terminal may not operate

correctly.

The HL register contains the new value of M.MODE.

The significance of each bit is as follows:

```
Bits        b8 b7 b6 b5 b4 b3 b2 b1
             0  0  M  M  M  S  S  E

Bit 1:        1 = erase screen, 0 = do not erase screen.
              (See section 3.2.3.1.2.4.4 for a
              description of the actions taken when the
              screen is erased.)
Bits 2 to 3:  00 -- mode inverse
              01 -- mode rewrite
              10 -- mode erase
              11 -- mode write
Bits 4 to 6:  current data processing mode.  This is
              valid only on CDC terminals.
```

At the completion of this routine, the data passed to it is stored in variable M.MODE. Before it is stored, the data is shifted right, shifting off the erase bit. The screen mode bits will be the least significant bits of the stored data.

```
Entry - HL = Data: Mode, screen mode, screen erase
Exit - M.MODE = Right shifted HL data.
```

4.2.12. R.STEPX (11/21/83)

This routine will step the X Coordinate one dot position. The direction of the step is dependant upon the direction bits set in M.DIR. The Y coordinate is not affected.

```
Entry - M.DIR set for desired direction by
        previously executed R.DIR.
Exit - X coordinate set according to the bit in
       M.DIR.
```

4.2.13. R.STEPY (11/21/83)

This routine will step the Y Coordinate one dot position. The direction of the step is dependant upon the direction bits set in M.DIR. The X coordinate is not affected.

```
Entry - M.DIR set for desired direction by
        previously executed R.DIR.
Exit - Y coordinate set according to M.DIR.
```

4.2.14. R.WE (11/21/83)

This routine will write or erase one dot position specified by the current coordinates.

```
Entry - M.MODE set for desired write or erase
        operation by previously executed R.MODE.
Exit - Dot at current (X,Y) location set according
```

4.2.15. R.DIR (11/03/83)

This routine sets the direction of the step routines. The lower 2 bits of HL register pair are referenced and stored for the X/Y directions as follows:

Bit 1: 0 = Step Y forward (up)
1 = Step Y reverse (down)
Bit 2: 0 = Step X forward (right)
1 = Step X reverse (left)

Entry - HL = Direction bits
Exit - M.DIR = Direction bits

4.2.16. R.INPUT (11/16/83)

This routine allows the Micro PLATO interpreter to get input from the keyboard and other devices. The variable M.KSW indicates what input devices send data where. When bit 1 is set to 1 keyboard and touch panel keys should be sent to Micro PLATO. When bit 1 is 0, these keys should be sent to the host. When bit 2 is set to 1, external keys (from external devices) should be sent to Micro PLATO, and when bit 2 is 0, these keys should be sent to the host. At no time should echo keys or unsolicited status words be sent to Micro PLATO.

Upon returning from R.INPUT the HL register pair contains the following:

	H REGISTER	L REGISTER
Keyboard	0 0 0 0 0 0 0 0	0 <--KEY DATA-->
Touch Panel	0 0 0 0 0 0 0 1	4 bit X 4 bit Y
External	0 0 0 0 0 0 1 0	8 bits ext. data

Each time R.INPUT is called, one word from the user buffer is retreived. If the most significant bit of H register is set to one, the buffer is empty and the data is invalid. Conversely, if the most significant bit is zero, the data is valid.

When key data is received, it is coded in the Micro PLATO key codes listed in Appendix D. These codes are not ASCII.

4.2.17. R.SSF (11/04/83)

This routine operates identically to the SSF command described in section 3.2.3.1.6. Upon entry the HL register pair must contain information identical to bits 01 through 15 of the SSF command format. If a read operation is selected, the data read from the device will be returned in the L register if M.KSW is set appropriately.

If a set interrupt mask operation is selected, a copy of the interrupt mask must be placed in the M.ENAB variable. Micro PLATO will use that information in subsequent calls to set the interrupt mask, so it must be maintained.

4.2.18. R.CCR (11/30/83)

This routine is used to establish character plotting parameters and select character sets (M0 to M7) to be used by the character generator. The character plotting information is passed to this routine as the contents of HL registers as follows:

BIT	FUNCTION
1	1 = vertical, 0 = horizontal
2 to 4	Selects character set 0 to 7
5	unused
6	1 = size two, 0 = size zero
7	1 = reverse, 0 = forward
8	Uncover code flag
9 to 16	Unused

Entry: HL is new value for M.CCR.
Exit: M.CCR set and other values initialized that depend on the values in this variable.

4.2.19. R.EXTOUT (11/03/83)

This routine outputs data to the device selected in R.SSF. The user stores data in memory and passes the address of the data in HL register pair, and the length in bytes in DE register pair.

Entry - HL = Origin (First Word Address)
DE = Length (Number of Bytes)

Exit - Data output to the device.

4.2.20. R.EXEC (11/08/83)

R.EXEC should process all the data that has built up in the buffer from the host while Micro PLATO has had control. It should also transmit any data that has built up in the buffer to the host, if there is any.

For simultaneous display of Micro PLATO and host output, the Micro PLATO interpreter will call R.EXEC periodically.

4.2.21. R.GJOB (11/03/83)

This routine will retrieve one byte from the host receive buffer and return it to the user program in the C register. These bytes are the raw ASCII codes received directly from the host.

This routine must not be called unless the variable M.JOBS is non-zero.

Entry - none
Exit - C register = data from host receive buffer

4.2.22. R.XJOB (11/03/83)

This routine 'executes' the byte in the C register as though it was received over the communications line. This might be used to monitor everything that is received from the host if used in conjunction with the R.GJOB routine.

Note that some commands and data units are longer than one byte, and therefore require a call to R.XJOB for each byte.

Also note that the terminal controlware resident will abort a command or data element in process upon the receipt of a control code.

Entry - C register = byte to be processed
Exit - none

4.2.23. R.RETURN (11/03/83)

This is an obsolete entry point which was used to return control to the resident after control was received by the resident during interrupt processing via the M.EXTPA interrupt handler. Interrupt routines should do a simple return instruction to their caller when complete.

4.2.24. R.CHRCV (11/16/83)

This routine is used to convert character set data in the eight word format used by the host and Micro PLATO into a form usable for character generation. The low byte of each word is the lower half of the vertical character strip, and the upper byte is the upper half of the strip, as described in section 3.2.3.1.2.3.

Entry - DE = First word address of character data.
HL = Number of characters to convert, each character consisting of 16 bytes.

On CDC terminals R.CHRCV should operate on the character set in place, at the address specified.

On other machines the character set should be translated into the resident's character set memory. The resident will be able to tell what character is being loaded by examining the MEM2 variable, which the Micro PLATO interpreter will set to the base of its character set. So, the character slot number is computed by ((DE) - (MEM2))/16 (the contents of DE minus the contents of MEM2 divided by 16). Micro PLATO assumes that M2 and M3 are contiguous, so characters in the range 0 to 63 are in M2 and characters in the range 64 to 127 are in M3.

4.2.25. R.ALARM (11/04/83)

This routine sounds the terminal's alarm.

4.2.26. R.PRINT (11/21/83)

The propose of R.PRINT is to allow Micro PLATO to initiate a

screen copy. The STOP or SHIFT-STOP keys will be active during the screen copy to allow the user to abort it before it is done.

There are no parameters to the R.PRINT routine. Upon exit status will be returned in the HL register as follows:

```
Exit: HL =
      0000 hex = screen copy completed, no errors.
      8000 hex = screen copy not available.
      8001 hex = printer not connected or turned off.
      8002 hex = printer not ready, out of paper.
```

4.2.27. R.FCOLOR (11/04/83)

This routine sets the foreground color to be used in subsequent graphics operations. The color is a 24-bit number passed in the the H, L and D registers. The color is maintained internally by Micro PLATO and the resident: there is no M. variable for color.

```
Entry:
      H: red intensity value
      L: green intensity value
      D: blue intensity value
```

4.2.28. R.BCOLOR (11/04/83)

This routine sets the background color to be used in subsequent graphics operations. The color is a 24-bit number passed in the the H, L and D registers.

```
Entry:
      H: red intensity value
      L: green intensity value
      D: blue intensity value
```

4.2.29. R.PAINT (11/21/83)

This routine initiates a paint operation, as described in section 3.2.3.1.8. There are no arguments or returned values. Only the solid fill option is currently availble through this entry point.

```
Entry - HL = 0 for solid fill
        HL = non-zero value for character fill
             when that feature is added. The format
             will be defined at a later date.
```

4.3. Terminal Common Variables (11/30/83)

The "M." variables are used to communicate between the resident and the Micro PLATO interpreter. Almost all of these variables should be changed only by calling the corresponding "R." entry point. This is because some variables, such as the current (X,Y) coordinates, cause other internal resident variables to change or hardware to be initialized. Certain variables, however, can be written directly by the Micro PLATO interpreter: M.EXTPA, M.MARGIN,

M.KSW, the mode 5, 6, and 7 addresses and the MEM2 alternate character set address. All variables may be read without restriction.

The addresses of the M. variables is given in the processor specific information in section C.

4.3.1. M.LDFILE (11/04/83)

This single byte variable contains the terminal resident load file number. The values are listed in 3.2.3.1.4.3.

Non-CDC terminals should place zero in this variable.

4.3.2. M.SBTYPE (11/04/83)

This eight-bit variable contains the terminal subtype. Its values are described in section 3.2.3.1.4.2.

4.3.3. M.CONFIG (11/30/83)

This single byte variable contains the terminal resident hardware configuration. See section 3.2.3.1.4.4 for the values it can contain.

4.3.4. M.M0 (11/04/83)

This two-byte variable contains the beginning address in memory of the M0 character set. The beginning address of M1 character set is 620 hex greater than the beginning address of M0. This is useful only in CDC terminals. Non-CDC terminals need not set this value.

4.3.5. M.VERS (04/12/83)

This location contains the version of the controlware resident that is loaded in the terminal memory.

4.3.6. M.TYPE (11/21/83)

This eight-bit variable contains the terminal type and the terminal configuration. All ASCII terminals must set this variable to 12 decimal.

The contents of M.TYPE are defined as follows for other terminals.

Bits 1-4	Meaning
0	Magnavox
1	PDP-11
2	IST-I, version B04, 4k memory
3	IST-I, version B05C, 8k memory
4	IST-I, version B05
5	IST-II & IST-III (without error detection)
6	IST-I, version B04, 8k memory
7	IST-I, version B04, 24k memory

8	PPT Serial Panel
9	PPT Old resident
A	PPT Standard version
B	TV
C	ASCII terminal
D	New TV
E	IST-I, version B006
F	IST-II and IST-III, (enhanced error)

All types are included for reference.

Bits 5 to 7 (terminal configuration)

CODE	IST-I MEMORY	IST-II IST-III MEMORY	IST-I IST-II TOUCH PANEL	IST-I IST-II VERSION
0	Unassigned	16K	no T. P.	Production
1	Unassigned	32K	no T. P.	Production
2	4K Memory	16K	With T. P.	Production
3	20K Memory	32K	With T. P.	Production
4	Unassigned	16K	no T. P.	Experimental
5	Unassigned	32K	no T. P.	Experimental
6	8K Memory	16K	With T. P.	Experimental
7	24K Memory	32K	With T. P.	Experimental

4.3.7. M.CLOCK (11/21/83)

This location is incremented once every 17.5 milliseconds on CDC terminals. Since it is a sixteen bit value it will wrap around at 65535 back to 0. The variable should be incremented by that value on other machines if possible, and if not it should be updated at some interval because Micro PLATO uses it as a random number seed.

4.3.8. M.EXTPA (11/21/83)

This sixteen bit variable holds the external processor address. When an interrupt for an external device occurs, this variable should be examined. If zero, the resident should read in a byte from the interrupting device, format it as an external key and send it to either the host or Micro PLATO depending on M.KSW. If M.EXTPA is non-zero, it is an address for the Micro PLATO interrupt handler. Control should be passed to the address that M.EXTPA points to. After Micro PLATO has performed its task, it will return control to the resident via a return instruction. The resident should then re-enable interrupts and perform any other post-interrupt processing required.

This variable is set by Micro PLATO directly.

4.3.9. M.MARGIN (11/04/83)

This sixteen bit variable contains the current margin. This margin should be used by the R.CHARS routine when a carriage return is peformed. The Micro PLATO interpreter will set this variable directly.

4.3.10. M.JOBS (11/21/83)

This one byte variable contains the number of bytes in the receive buffer. A zero value indicates the buffer is empty. The full value will depend upon the terminal, because the size of the receive buffer may vary between terminal types.

Since this is a single byte value, terminals with buffers larger than 256 bytes should put 255 in this location when there are 255 or more characters in the receive buffer.

4.3.11. M.CCR (11/04/83)

This single byte variable contains the character plotting parameters. R.CCR must be called to set this variable. The contents of M.CCR are:

Bit	Description
1	Vertical plotting when = to one, horizontal when = zero.
2, 3, 4	Set to 0 through 7 for selected character set M0 through M7 (Micro PLATO character set numbering)
5	Unused
6	1 = size 2 characters, 0 = size 0.
7	1 = reverse plotting, 0 = forward.
8	Uncover code flag, used by R.CHARS.

4.3.12. M.MODE (11/21/83)

This variable contains the current terminal operating mode and the screen mode. R.MODE must be called to set this variable. The format of M.MODE is the same as the parameter passed to R.MODE except that M.MODE is shifted right 1 bit position.

The contents of M.MODE are:

Bit Description

Screen mode bits

1	2	Mode 3	All Other Modes
0	0	Inverse	Erase
0	1	Rewrite	Write
1	0	Erase	Erase
1	1	Write	Write

5	4	3	Processing mode (CDC terminals)
0	0	0	Point Plot
0	0	1	Draw Line
0	1	0	Load Memory
0	1	1	Character Plot
1	0	0	Block Write/Erase
1	0	1	User Program
1	1	0	User Program
1	1	1	User Program

8	7	6	These bits should be zero.

4.3.13. M.DIR (11/04/83)

This variable contains flags to indicate the direction of movement for both the X and Y coordinates when the user calls routine R.STEPX or R.STEPY. R.DIR must be called to set this variable.

The contents of M.DIR are as follows:

Bit	Description
1	0 = increment Y coordinate. 1 = decrement Y coordinate.
2	0 = increment X coordinate 1 = decrement X coordinate
3 to 8	Not used.

4.3.14. M.KSW (11/04/83)

This variable determines whether the data from the keyboard, touch panel, or external device are to be passed to the host or to Micro PLATO. The variable must be set directly by Micro PLATO. See sections 3.2.3.4 and 4.2.16 for further discussion.

If bit 1 of M.KSW is 1 keyboard and touch panel input should be sent to Micro PLATO. If it is 0, then data from those sources is sent to the host.

If bit 2 of M.KSW is 1 external data should be sent to Micro PLATO, otherwise it should be formatted as external keys and sent to the host.

4.3.15. M.ENAB (11/04/83)

This variable contains the internal interrupt mask. It must be set by calling R.SSF.

The contents of M.ENAB are as follows:

Bit	Description
1	Unassigned
2	Unassigned
3	Unassigned
4	1 = Enable external channels interrupt 0 = Disable external channels interrupt
5	Always one
6	1 = Enable touch panel 0 = Disable touch panel
7	Always one
8	Always one

4.3.16. Mode 5, 6 and 7 addresses (11/30/83)

These sixteen bit variables are set by Micro PLATO directly. When the resident receives data from the host in modes 5, 6 or 7, the corresponding address (MOD5A, MOD6A or MOD7A) should be examined. If non-zero, control should be transferred to the address pointed to and the data from the host passed in the appropriate registers (see section 3.2.3.1.2.6).

4.3.17. Character memory addresses (11/04/83)

These variables contain the addresses of the character set memories. Only MEM2 and MEM3 are used currently. When Micro PLATO loads a character set these variables will be loaded with the base addresses of the memories involved. On CDC terminals the character sets will be stored and converted in place at these addresses. On other machines, the character sets are copied and converted to an internal representation inside the resident.

4.3.18. M.HALT (11/04/83)

This variable is used to stop execution of programs running in the terminal. Such programs, called PPT programs (for Programmable Plato Terminal), must monitor this location themselves to determine whether they should halt. The values of the bits for M.HALT are:

Bits 8 to 3: 0
Bit 2: Independant program stop bit.
Bit 1: Dependant program stop bit.

When lesson-dependant programs find the dependant program stop bit set, they should stop execution and return to the resident. Similarly, when lesson-independant programs find the independant program stop bit set, they should return to the resident. The host will write into this variable with load memory commands. Thus, only CDC terminals will support this option. In fact, the residents are oblivious to all this, except that they provide the low-level services to the PPT programs and the host.

4.3.19. M.STATUS (11/04/83)

This variable is used to control the execution of programs running in the terminal from the host. The values of the bits and their functions are:

Bit 8 = 0
Bit 7 = 0
Bit 6 = 1
Bit 5 = Complement of the independant program bit
Bit 4 = Complement of the dependant program bit
Bit 3 = 0
Bit 2 = Independant program bit
Bit 1 = Dependant program bit

If a lesson-dependant program is running in the terminal the dependant bit is set to one. If a lesson independant program is running, then the independant bit is set to one. The host is able to query this variable to determine the

status of programs running in the terminal.

Note: the resident is not involved in this process at all, except that it serves as an intermediary, providing the program with data from the host via mode 5, 6 and 7, and sending data to the host for the program via the R.XMIT entry point.

5. Appendix A: PLATO character set (ii/i6/83)

CHARACTER SET M0

Low nibble	High nibble 00	i0	20	30	40	50	60	70	
0	NUL	DLE	SP	0	@	P	#q	p	
1	SOH	DC1	!	1	A	Q	a	q	
2	STX	DC2	"	2	B	R	b	r	
3	ETX	DC3	#	3	C	S	c	s	
4	EOT	DC4	$	4	D	T	d	t	
5	ENQ	NAK	%	5	E	U	e	u	
6	ACK	SYN	&	6	F	V	f	v	
7	BEL	ETB	'	7	G	W	g	w	
8	BS	CAN	(	8	H	X	h	x	
9	HT	EM	)	9	I	Y	i	y	
a	LF	SUB	*	:	J	Z	j	z	
b	VT	ESC	+	;	K	[	k	{	
c	FF	FS	,	<	L	\	l	¦	
d	CR	GS	-	=	M	]	m	}	
e	SO	RS	.	>	N	#x	n		#n
f	SI	US	/	?	O	_	o	del	

PLATO terminal resident specification

CHARACTER SET M1

20	SP	30	#a	40	c _ _#u (umlaut)
21	/	31	#b	41	#O
22	=	32	#d	4243	o
23	#n	33	#1	44	#F
24	@	34	#m	45	*
25	<>	35	#p	46	#e (acute accent)
26	#W	36	#r	47	#c (cedilla)
27	->	37	#s	48	#v (hac#vek)
28	#X	38	#w	49	,
29	<-	39	<=	4a	!
2a	*	3a	>=		
2b	+	3b	#t		
2c	-	3c	<		
2d	u	3d	#o		
2e	^	3e	>		
2f	/	3f	>		

Notes: character 40 should combine with the left and right parentheses of the M0 character set to form the copyright symbol: (c). The cedilla (#c) should combine with c to form c#c correctly. The other diacritics (#e, #q, #n, #u, #v and #x) should combine with other letters: e#e, a#q, n#n, u#u, s#v, o#x.

6. Appendix B: Protocol summary (11/30/83)

This appendix contains a summary of the PLATO host control codes and escape sequences.

6.1. Control Codes (11/04/83)

CONTROL CHARACTER	HEX CODE	
BS	08	Backspace. Moves one character width back.
HT	09	Tab. Moves one character width.
LF	0A	Linefeed. Moves one character height down.
VT	0B	Vertical tab. Moves one character height up.
FF	0C	Form feed. Set (X,Y) position to top of page.
CR	0D	Carriage return. Set (X,Y) to margin on next line.
EM	19	Selects block write/erase mode (mode 4).
ESC	1B	Causes terminal to examine the next character to determine if it is a valid escape sequence.
FS	1C	Selects point-plot mode (mode 0).
GS	1D	Selects draw line mode (mode 1).
US	1F	Selects alpha mode (mode 3).

Note: control codes received not included here should be ignored.

6.2. Escape Sequences (11/01/83)

ESC SOH	Used to select Tektronix emulation mode in CDC IST III graphics and in the Viking 721-30 residents.
ESC STX	Select PLATO operation.
ESC ETX	Select TTY operation.
ESC FF	Clear screen and set (X,Y) to top of page.
ESC DC1	Select inverse video screen mode.
ESC DC2	Select mode write.
ESC DC3	Select mode erase.
ESC DC4	Select mode rewrite.
ESC ESC	This has no function, but the resident must still recognize a character coming after two escapes as a valid escape sequence (that is, ESC ESC FF should be interpreted as ESC FF).
ESC 2	Load Coordinate command
ESC @	Superscript
ESC A	Subscript
ESC B	Select Character set M0.
ESC C	Select Character set M1.
ESC D	Select Character set M2.
ESC E	Select Character set M3.
ESC F	Select Character set M4.
ESC G	Select Character set M5.
ESC H	Select Character set M6.
ESC I	Select Character set M7.
ESC J	Select horizontal.
ESC K	Select vertical.
ESC L	Select forward.
ESC M	Select reverse.

ESC N	Select size 0.
ESC O	Select size 2.
ESC P	Load memory (mode 2 data with character convert).
ESC Q	SSF command.
ESC R	External data.
ESC S	Load memory (mode 2 data).
ESC T	Select mode 5.
ESC U	Select mode 6.
ESC V	Select mode 7.
ESC W	Load Memory Address command.
ESC Y	Load Echo command.
ESC Z	Set Margin.
ESC a	Set foreground color.
ESC b	Set background color.
ESC c	Paint.

Note: Escape sequences not shown have no effect on the terminal.

7. Appendix C: Processor information (11/21/83)

This appendix gives the mapping of the Z80 register set to that of other processors. It also enumerates assumptions made about the environment of Micro PLATO.

7.1. Z80 (11/08/83)

7.1.1. Register mapping (11/30/83)

The Z80 registers all map to themselves. When the resident is called, it must preserve the IX and IY registers. All other registers may be changed.

7.1.2. M. and R. addresses (11/08/83)

The R. table consists of a sequence of jump instructions to the appropriate routines, so each entry is three bytes long.

Hex Address	Label
0040	R.INIT
0043	R.DOT
0046	R.LINE
0049	R.CHARS
004C	R.BLOCK
004F	R.INPX
0052	R.INPY
0055	R.OUTX
0058	R.OUTY
005B	R.XMIT
005E	R.MODE
0061	R.STEPX
0064	R.STEPY
0067	R.WE
006A	R.DIR
006D	R.INPUT
0070	R.SSF
0073	R.CCR
0076	R.EXTOUT
0079	R.EXEC
007C	R.GJOB
007F	R.XJOB
0082	R.RETURN
0085	R.CHRCV
0088	R.ALARM
008B	R.PRINT

The following table gives the offsets of the M. variables for the Z80 processor.

Hex Address	Label	Description
22E5	M.LDFILE	Controlware load file number
22E6	M.SBTYPE	Terminal Subtype
22E7	M.CONFIG	Terminal Configuration

22E8	M.M0	Location of M0 character set
22EA	M.VERS	Controlware version
22EB	M.TYPE	Terminal Type
22EC	M.CLOCK	Real-time Clock
22EE	M.EXTPA	Interrupt Processor Address
22F0	M.MARGIN	Margin used for Carriage Returns
22F2	M.JOBS	Job Stack Count
22F4	M.CCR	Character Plotting Parameters
22F6	M.MODE	Current Terminal Operating Mode
22F8	M.DIR	Direction of Coordinate Step
22FA	M.KSW	Device Input Director
22FC	M.ENAB	Internal Interrupt Mask
22FE	unused	
2300	MOD5A	Mode 5 Entry Address
2302	MOD6A	Mode 6 Entry Address
2304	MOD7A	Mode 7 Entry Address
2306	M2	Character Set M2 First Word Address
2308	M3	Character Set M3 First Word Address
230A	M4	Character Set M4 First Word Address
230C	M5	Character Set M5 First Word Address
230E	M6	Character Set M6 First Word Address
2310	M7	Character Set M7 First Word Address
2314	M.HALT	Turns off user program
2316	M.STATUS	Status of usert program

7.1.3. Micro PLATO interpreter loading (11/11/83)

On CDC terminals the boot ROM reads in a control program from the disk drive. This control program determines the type of the terminal (IST III, IST II or Viking) and reads in the appropriate resident. The Micro PLATO interpreter is also read into memory by this program. Control is passed to different locations in the resident, depending on whether the resident was loaded from disk or from the host. If the resident was loaded from disk, initializations are performed and control is passed to the Micro PLATO interpreter whose address is known to the resident. If the resident was loaded from the host, it goes into TTY mode immediately.

7.2. 8086 (11/21/83)

This section describes Micro PLATO under the MSDOS and derivative operating systems using an 8086 or compatible microprocessor.

7.2.1. Register mapping (11/04/83)

On the 8086 and compatible processors (8088, 80186, 80188 and 80286) Micro PLATO must be in a single segment (that is, the DS, ES and CS registers are set to the same value). Micro PLATO does not use the stack explicitly except for pushing and popping, so it may reside anywhere except in Micro PLATO's segment.

Micro PLATO uses all the registers (including BP). Micro PLATO assumes that SI and DI are unchanged between calls to the resident. Any other registers may be modified.

The correspondence used between Z80 and 8086 registers is:

Z80	8086
A	AL
H	BH
L	BL
D	DH
E	DL
B	CH
C	CL
IX	SI
IY	DI

7.2.2. M. and R. addresses (11/23/83)

The Micro PLATO interpreter performs only short calls. The R. entry points must be far calls followed by short returns. Each jump table entry is then six bytes. The table below gives the offsets for the R. routines. (All values in hex.)

```
r.init   equ    0000
r.dot    equ    0006
r.line   equ    000c
r.chars  equ    0012
r.block  equ    0018
r.inpx   equ    001e
r.inpy   equ    0024
r.outx   equ    002a
r.outy   equ    0030
r.xmit   equ    0036
r.mode   equ    003c
r.stepx  equ    0042
r.stepy  equ    0048
r.we     equ    004e
r.dir    equ    0054
r.input  equ    005a
r.ssf    equ    0060
r.ccr    equ    0066
r.exto   equ    006c
r.exec   equ    0072
r.gjob   equ    0078
r.xjob   equ    007e
r.return equ    0084
r.chrcv  equ    008a
r.alarm  equ    0090
r.print  equ    0096
r.fcolor equ    009c
r.bcolor equ    00a2
r.paint  equ    00a8
r.dummy1 equ    00ae
r.dummy2 equ    00b4
r.dummy3 equ    00ba
```

(The entry points r.dummy1, r.dummy2 and r.dummy3 are reserved for future use. They should simply return.)

The M. table for the 8086 is as follows:

```
m.ldfile equ    00c0
m.sbtype equ    00c1
m.config equ    00c2
m0       equ    00c3
m.vers   equ    00c5
```

```
m.type    equ     00c6
m.clock   equ     00c7
m.extpa   equ     00c9
m.margin  equ     00cb
m.jobs    equ     00cd
m.ccr     equ     00cf
m.mode    equ     00d1
m.dir     equ     00d3
m.ksw     equ     00d5
m.enab    equ     00d7
mod5a     equ     00db
mod6a     equ     00dd
mod7a     equ     00df
m2        equ     00e1
m3        equ     00e3
m4        equ     00e5
m5        equ     00e7
m6        equ     00e9
m7        equ     00eb
m.halt    equ     00ed
m.status  equ     00ef
```

7.2.3. Micro PLATO interpreter loading (11/21/83)

Micro PLATO must be loaded in its own segment. Under MSDOS the interpreter is in an absolute binary form and must be loaded into its segment at location 0000. The entirety of the interpreter file (called MPLATO.SYS) must be loaded at this location.

The M. variables and the R. jump table are placed in the lower part of the Micro PLATO segment. They should be initialized to the correct values and then control should be passed to the interpreter at location 00f7 hex. The address of the R.INIT entry point should be pushed on the stack before giving control to Micro PLATO, so that a short RET instruction will give control back to the resident.

When Micro PLATO is executed register BX must contain the highest address available for the interpreter in the segment. If an entire 64K segment is available, BX should contain 0FFFF hex. The minimum segment size for Micro PLATO is 32K. If there is insufficient memory the resident should not even try to load the interpreter.

8. Appendix D: PLATO keycodes (11/08/83)

PLATO KEY CHARACTERS AND CODES

PLATO KEY SYMBOL		MICRO PLATO KEY CODE (HEX) SHIFT		ASCII CHAR. GENERATED SYMBOL		ASCII CHAR. CODE (HEX) SHIFT	
LOWER	UPPER	OFF	ON	LOWER	UPPER	OFF	ON
a	A	41	61	a	A	61	41
b	B	42	62	b	B	62	42
c	C	43	63	c	C	63	43
d	D	44	64	d	D	64	44
e	E	45	65	e	E	65	45
f	F	46	66	f	F	66	46
g	G	47	67	g	G	67	47
h	H	48	68	h	H	68	48
i	I	49	69	i	I	69	49
j	J	4A	6A	j	J	6A	4A
k	K	4B	6B	k	K	6B	4B
l	L	4C	6C	l	L	6C	4C
m	M	4D	6D	m	M	6D	4D
n	N	4E	6E	n	N	6E	4E
o	O	4F	6F	o	O	6F	4F
p	P	50	70	p	P	70	50
q	Q	51	71	q	Q	71	51
r	R	52	72	r	R	72	52
s	S	53	73	s	S	73	53
t	T	54	74	t	T	74	54
u	U	55	75	u	U	75	55
v	V	56	76	v	V	76	56
w	W	57	77	w	W	77	57
x	X	58	78	x	X	78	58
y	Y	59	79	y	Y	79	59
z	Z	5A	7A	z	Z	7A	5A
0	<	00	20	0	<	30	3C
1	>	01	21	1	>	31	3E
2	[	02	22	2	[	32	5B
3	]	03	23	3	]	33	5D
4	$	04	24	4	$	34	24
5	%	05	25	5	%	35	25
6	_	06	26	6	_	36	5F
7	'	07	27	7	:	37	7C
8	*	08	28	8	*	38	2A
9	(	09	29	9	(	39	28
=	)	5B	7B	=	)	3D	29
+	+	0E	2E	+	#	2B	23
@	shift	0D	2D	#x	\	5E	5C
-	-	0B	2F	-	#n	2D	7E
/	^	0B	2B	-	'	60	27
*	u	0A	2A	&	@	26	40
;	:	5C	7C	;	:	3B	3A
.	!	5E	7E	.	!	2E	21
,	"	5F	7F	,	"	2C	22
/	?	5D	7D	/	?	2F	3F
SUPER	SUPER1	10	30	DC3	ETB	13	17
SUB	SUB1	11	31	EOT	ENQ	04	05
ANS	TERM	12	32	BEL	DC4	07	14
COPY	COPY1	1B	3B	ETX	SYN	03	16
TAB	CR	0C	2C	LF	FS	0A	1C
ERASE	ERASE1	13	33	BS	EM	08	19
MICRO	FONT	14	34	{	DEL	7B	7F

PLATO terminal resident specification

HELP	HELP1	15	35	VT	HT	0B	09
SQUARE	ACCESS	1C	3C	}	NULL	7D	00
NEXT	NEXT1	16	36	CR	RS	0D	1E
EDIT	EDIT1	17	37	SUB	CAN	1A	18
BACK	BACK1	18	38	STX	SO	02	0E
LAB	LAB1	1D	3D	FF	SI	0C	0F
DATA	DATA1	19	39	DC2	GS	12	1D
STOP	STOP1	1A	3A	SOH	DC1	01	11
SPACE	BACKSP	40	60	SPACE	US	20	1F
PRINT	PRINT1	1F	3F	See note 2.			

Notes:

1. The CNTL key is not used in PLATO operations.
2. The print key is used to initiate local printing in the terminal when appropriate printer is attached. It is not transmitted to the PLATO host.

9. Appendix E: Existing terminals (11/01/83)

This appendix gives the characteristics of the terminals that have been adapted to use the PLATO ASCII communications protocol.

9.1. CDC terminals (11/01/83)

All Control Data terminals follow the terminal model. The screen resolution is 512 by 512 with a square aspect ratio.

The keyboards all have the special PLATO keys labeled on them.

Every terminal has a clear plastic touch panel for pointing.

All are monochrome, with the ISTs using black and white monitors and the Viking V721-30 using a green screen.

There is one serial port available for additional I/O on the IST III. Two are available on the Viking. All CDC terminals have a parallel interface that allow connection to intelligent disk drives and other peripherals.

Screen printing is available on all the terminals with the Viking V721-30 and the IST III for both the CDC 726-10 and 726-20 graphics printers, while screen printing on the IST I and IST II is supported for the 726-10 only.

9.2. Zenith Z100 (11/01/83)

The Zenith Z100 has two separate screen resolutions, selectable from the keyboard and by a configuration file. The first is 256v by 512h, giving an 8 by 8 character size. The loadable character set is mapped from 16 by 8 to 8 by 8 by oring every two lines into one. The second resolution is 480v by 512h. A high-persistence monitor is required to minimize flicker in this mode. A 15 by 8 character size is supported. The bottom line of the 16 high character is omitted.

The Z100 has many function keys that are used to map the PLATO function keys and some of the special graphics characters (divide, multiply, etc.).

The Z100 has eight colors (black, white, red, green, blue, magenta, cyan and yellow).

A transparent plastic touch panel connected to a serial port is used as an auxiliary pointing device.

Other serial ports are configured to work as other I/O devices, and are able to generate interrupts.

Screen printing is available for the Epson series of printers through the standard printer port configured by the operating system.

9.3. IBM PC (11/01/83)

The screen resolution of the IBM PC is 256v by 512h, which is accomplished by reprogramming the CRT controller chip in the graphics adapter.

Character sets are 8 by 8, and alternate characters are mapped into physical characters in the same fashion as the Z100.

The function keys (F1 through F10) are used for PLATO function keys. In addition, some of the other keys are usurped from their usual function and made special PLATO keys (for example, escape is the assignment arrow).

No touch panel or other device is currently implemented.

The screen is monochrome only.

Screen printing is available for the Epson series (including the IBM system printer) on the parallel printer channel only.

9.4. Apple II (11/21/83)

The screen resolution for an Apple II is 192v by 256h centered in the Apple's true horizontal of 280 dots. No special reprogramming is done since the Apple does not have a programmable crt controller.

Character sets are 4 by 6. When alternate character sets are loaded, only the lower 4 by 6 of the character is kept. It was felt that it would be better for a human to design a new character set rather than to do the large amount of compression that would be needed to reduce 8 by 16 to 4 by 6.

Function keys are mapped onto the letter keys and prefixed with an ESCAPE. Pressing ESCAPE twice takes you to a built-in help page that can be used to find all key equivalents.

The touch panel is simulated through the use of game paddles or joystick. Again, the built-in help display explains the procedure.

For the communications drivers, if there are external switches or setup programs, the device should be set to 300 baud, 7 data bits, 1 stop bit, and even parity. For printers, the user's normal setup should work just fine except any "auto linefeed" option should be turned off. Anything that is specific to a particular card is listed below.

The only interface cards that should be used at 1200 baud are cards that can generate interrupts. Currently only the CCS 7710A and the Apple Super Serial Card can generate interrupts without hardware modification to the interface card.

Communications drivers:

* DC Hayes Micromodem II
* Mountain Computer CPS Card
* California Comp. Sys. 7710A Serial Card

* Apple Communications Card
* Apple Super Serial Card
  The "allow interrupts" switch must be set to "on".
* Prometheus VERSAcard

RS-232 modems for serial cards:
* General "dumb" modem
* DC Hayes Smartmodem

Screen prints in 2 sizes can be obtained. A list of supported printers is below.
Printers:
* Epson MX70/80/100 (with dot graphics)
  * Apple Parallel Card
  * Apple Communications Card
  * Mountain CPS Card
    Must have printer connected to parallel port.
  * Prometheus Versacard
  * Apple Super Serial Card
  * Apple Serial Card
* NEC PC8023-A
  * Apple Parallel Card
  * Apple Communications Card
  * Mountain CPS Card
    Must have printer connected to parallel port.
  * Prometheus Versacard
  * Apple Super Serial Card
  * Apple Serial Card
* Apple Silentype

9.5. Atari (11/08/83)

Two screen resolutions are supported by the Atari resident: a 'scrunch' mode with 320h by 192v and 5 by 6 characters, and a 'scroll' mode with 512h by 384v and 8 by 12 characters. In scrunch mode the entire screen is visible at once, while in scroll mode the joystick allows the user to scroll a window through the larger screen area, allowing about a quarter of the screen to be seen at once in the higher resolution.

Two copies of downloaded character sets are kept. Both are reduced to the appropriate size for the screen. Image processing techniques are used to preserve the quality of the character in the smaller grid size.

The function keys are handled by pressing the OPTION key and a mnemonic letter (shifted if required). For example, STOP is represented by OPTION S, and shift-DATA is entered as OPTION SHIFT D.

If in scrunch mode, the touch panel becomes active when the host sends down the appropriate SSF command. A cursor appears on the screen and the user directs the cursor to

the desired location. To make the touch, the button on the joystick is pressed.

The display is two colors only, but these colors are selected by the user for the best display.

A character-oriented screen print facility is planned (that is, only text will be printed when a print is initiated -- dot graphics will not be supported).

10. Appendix F: Serial channel control (11/11/83)

The serial channel is accessed through SSF and EXT commands, either from the host or from Micro PLATO (see section 3.2.3.1.6). Reading from device address 2 will read data from the currently selected serial channel. Writing to device address 2 will write data to the serial channel. Reading from device address 3 will return the current status of the selected serial device. Writing to device 3 will set up serial channel control information. Device address 4 selects the serial channel.

The following sections describe the formats for status and control for the various terminals.

10.1. Generic terminals (11/11/83)

All terminals but the IST III use the following scheme for controlling the serial ports (including the Viking 721-30).

10.1.1. Control (11/11/83)

To control the serial port, bytes are sent to device address 3. Each eight-bit byte received changes a certain parameter. The top four bits of the byte indicate which parameter is being set, while the lower four bits indicate the value of the parameter.

The following commands are available for controlling the serial port:

Set interrupt mask:

```
b8 b7 b6 b5   b4 b3 b2 b1
 0  0  0  0    1  1 RD RQ

RD
   0: Disable character ready interrupt
   1: Enable character ready interrupt
RQ
   0: Disable character request interrupt
   1: Enable character request interrupt
```

Set baud rate:

```
b8 b7 b6 b5   b4 b3 b2 b1
 0  0  0  1   r4 r3 r2 r1

r4, r3, r2, r1:
   0 0 0 0:     75 bps
   0 0 0 1:    110 bps
   0 0 1 0:    150 bps
   0 0 1 1:    200 bps
   0 1 0 0:    300 bps
   0 1 0 1:    600 bps
   0 1 1 0:   1200 bps
   0 1 1 1:   1800 bps
   1 0 0 0:   2400 bps
   1 0 0 1:   4800 bps
   1 0 1 0:   9600 bps
```

```
                    1 0 1 1: 19200 bps
                    1 1 0 0: ----
                    1 1 0 1: ----
                    1 1 1 1: ----
```

Select word length:

```
     b8 b7 b6 b5   b4 b3 b2 b1
      0  0  1  0    X  S L2 L1
                   X:
                      Don't care -- can be any value
                   S:
                      0: one stop bit
                      1: two stop bits
                   L2, L1:
                      0 0: 5 data bits
                      0 1: 6 data bits
                      1 0: 7 data bits
                      1 1: 8 data bits
```

Select parity:

```
     b8 b7 b6 b5   b4 b3 b2 b1
      0  0  1  1    X  X P2 P1

                   P1:
                      0: disable parity
                      1: enable parity
                   P2:
                      0: odd parity
                      1: even parity

                      (If parity is disabled, the state of
                      this bit does not matter.)
```

Modem control:

```
     b8 b7 b6 b5   b4 b3 b2 b1
      0  1  0  0    X  B  R  D

                   X:
                      Don't care
                   B:
                      Turn on break (space on line)
                   R:
                      Set request to send (RTS)
                   D:
                      Set data terminal ready (DTR)
```

10.1.2. Status (11/11/83)

A single byte of status is returned when a read from device address 3 is performed. Its format is:

```
     b8  b7  b6  b5  b4  b3  b2  b1
     CD  RI  DSR CTS RDE TXR BRK RXR

                   RXR: Receive ready (a character is ready
                        to receive from the serial channel)
```

BRK: Break detected on line
TXR: Transmit ready (the serial channel is ready to transmit another character)
RDE: Receive data error. An error occurred in receiving a character. This could be due to overrun or parity error.
CTS: clear to send.
DSR: data set ready
RI: ring indicator
CD: carrier detect

10.2. IST III serial channel (11/11/83)

Only one serial channel is available on the IST III, at device addresses 2 and 3. Writing to device address 4 will have no effect on this terminal.

10.2.1. Control (11/11/83)

Controlling the IST III's serial channel requires a different method. Each parameter is set by a sequence of two bytes rather than by a single byte. The sequences required for initializing the serial channel are as follows:

Byte 0	Byte 1	Function
00000000	00011000	Resets the serial channel
00000010	CCC00000	Sets baud rate
00000100	0100SSPP	Sets stop bits and parity
00000011	BB000001	Set receive bits/character
00000101	DBBZ10R0	Sets transmit bits/ character, DTR, break and RTS control.

The meanings of the letters are:

CCC	Baud rate
000	150 bps
001	300 bps
010	600 bps
011	1200 bps
100	2400 bps
101	4800 bps
110	9600 bps
111	19200 bps

SS	Number of stop bits
00	Do not use this.
01	1 stop bit
10	1 1/2 stop bits
11	2 stop bits

BB	Bits per receive character
00	5
01	7
10	6 (not a typo -- 6 and 7 are this way)
11	8

PP	Parity
X0	None

```
01              Odd
11              Even

Z               Break
0               Off
1               On

D               DTR
0               Off
1               On

R               RTS
0               Off
1               On
```

10.2.2. Status (11/11/83)

The status on the IST III can be read from device address 3. The meanings of the bits in the returned byte are:

```
b8  b7  b6  b5  b4  b3  b2  b1
BRK NDF CTS ERR DCD TXR DSR RXR

RXR = 1, receive character ready
DSR = 1, data set ready
TXR = 1, ready to transmit character
DCD = 1, data carrier detect
ERR = 1, receive error occurred
CTS = 1, clear to send
NDF = not defined, will be zero
BRK = 1, break received
```

****

Print of s0ascers/pea completed 12/20/83 at 11:32 am

67 pages, 3393 lines printed in 32 minutes 4 seconds

TIPS = 7.2
DAPM = 8.4

****